

TESIS DE LA UNIVERSIDAD
DE ZARAGOZA

2021

277

Íñigo Alonso Ruiz

Semantic Segmentation for Real-World Applications

Director/es

Montesano del Campo, Luis
Murillo Arnal, Ana Cristina

<http://zaguan.unizar.es/collection/Tesis>

ISSN 2254-7606



Premsas de la Universidad
Universidad Zaragoza

© Universidad de Zaragoza
Servicio de Publicaciones

ISSN 2254-7606



Universidad
Zaragoza

Tesis Doctoral

SEMANTIC SEGMENTATION FOR REAL-WORLD
APPLICATIONS

Autor

Íñigo Alonso Ruiz

Director/es

Montesano del Campo, Luis
Murillo Arnal, Ana Cristina

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2021



Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza



PhD Thesis

Semantic Segmentation for Real-World Applications

Autor

Íñigo Alonso Ruiz

Directores

Ana Cristina Murillo Arnal

Luis Montesano del Campo

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza, 2021

Acknowledgments

We are a product of the decisions we make, the people we meet, and the circumstances around us. I feel really grateful and lucky for having the means that allowed me to be what I am now.

First of all, I would like to thank Ana Cristina for introducing me to this world and for all the support, help, and knowledge she gave me during this thesis. She and her family (the great teacher and person Darío and their cute son Diego). Thanks a lot also to Luis Montesano, especially for all the comments during the PhD. Your help and directions especially in our meetings and your contribution to our works were really valuable. Thanks to all the other professors that also contributed to my growth like Luis Riazuelo (the coolest Lab boss), Danilo (with his humor and investing knowledge), Eduardo (with his great *control* of everything) and other lab colleges and teachers. This university has really good professors, even those that do not believe in deep learning.

Thanks to all the PhD peers for all the good moments: Chema, the boss, since I know you from the bachelor I really you up to you man, I wish you all the possible luck. León, you are family, you know. Alberto, thanks for all the interesting conversations, I pay a lot of *attention* to them, they *transformed* me hehe. Clara, thanks for being so nice. I really enjoyed all the time we had together. Being such a nice person will only bring you nice things in your life. Manu, thanks for those meals. Berta and Carlos, apart from all the help you gave me, I really remember with special care our trip to Montreal. Pablo and Sara, I am going to miss our times eating in the lab. Jose, you are one of the nicest persons I know. To all the rest, Richard, Seong, Leo, Rafa, thank you. If you are not in this list, please, cite my works so that I can remember acknowledge you in my next thesis. Just kidding (you can cite my works though). I really enjoyed all the moments and the people during this thesis. Thanks to everyone that spend some of their valuable time to be with me. You all are part of me now, Thanks.

Thanks to my both parents that provided me with all I needed during all my life. Thanks to my close friends outside the university for all the esteem and regard I felt from you during these years (only mentioning the closest otherwise I would not ever end) Marcos Raga, Ruben Moreno, and Jorge Cancer. Thanks for listening, for being there, for making me just forget about meaningless things, and make me enjoy life. I hope we can be together for a long time. Thanks for my special and closest colleges during my three internships: Ignacio Alzugaray thanks for helping me surviving in Zurich, for all the attention and help you gave me, for introducing me to other colleges, and for everything you did for me during my internship at the ETH. Kamil thanks for the funny conversations when I was in Munich, when I was with you, I was the real Iñigo. Thanks for all the happy moments we had and we still have. Marco thanks for all the things I learned from you during my internship, my Italian friend. Andres, thanks for being my Spanish guy in Amazon, you really cared for me and help me a lot. Thanks for being still there and be a supporting friend. David, I also learned a lot from you. Thanks for all the paper-sharing we had and still have. Thanks for all the opportunities you opened me and thanks for having been the nicest mentor ever. I learned a lot from you, both professional and personal things.

And last, but not least, special thanks to Lara for standing with me during the last seven years and for listening to all my complaints about the dark sides of the PhD life. Sushi tastes better when eaten with you.

We should not forget we are just a speck of life and time. We are not better than others. We are just a result of uncontrollable and random factors and circumstances, influenced by people around us. And probably a result of our decisions.

Resumen

En visión por computador, la comprensión de escenas tiene como objetivo extraer información útil de una escena a partir de datos de sensores. Por ejemplo, puede clasificar toda la imagen en una categoría particular o identificar elementos importantes dentro de ella. En este contexto general, la segmentación semántica proporciona una etiqueta semántica a cada elemento de los datos sin procesar, por ejemplo, a todos los píxeles de la imagen o, a todos los puntos de la nube de puntos. Esta información es esencial para muchas aplicaciones de visión por computador, como conducción, aplicaciones médicas o robóticas. Proporciona a los ordenadores una comprensión sobre el entorno que es necesaria para tomar decisiones autónomas.

El estado del arte actual de la segmentación semántica está liderado por métodos de aprendizaje profundo supervisados. Sin embargo, las condiciones del mundo real presentan varias restricciones para la aplicación de estos modelos de segmentación semántica. Esta tesis aborda varios de estos desafíos: 1) la cantidad limitada de datos etiquetados disponibles para entrenar modelos de aprendizaje profundo, 2) las restricciones de tiempo y computación presentes en aplicaciones en tiempo real y/o en sistemas con poder computacional limitado, y 3) la capacidad de realizar una segmentación semántica cuando se trata de sensores distintos de la cámara RGB estándar. Las aportaciones principales en esta tesis son las siguientes:

1. Un método nuevo para abordar el problema de los datos anotados limitados para entrenar *modelos de segmentación semántica a partir de anotaciones dispersas*. Los modelos de aprendizaje profundo totalmente supervisados lideran el estado del arte, pero mostramos cómo entrenarlos usando solo unos pocos píxeles etiquetados. Nuestro enfoque obtiene un rendimiento similar al de los modelos entrenados con imágenes completamente etiquetadas. Demostramos la relevancia de esta técnica en escenarios de monitorización ambiental y en dominios más generales.

2. También tratando con datos de entrenamiento limitados, proponemos un método nuevo para *segmentación semántica semi-supervisada*, es decir, cuando solo hay una pequeña cantidad de imágenes completamente etiquetadas y un gran conjunto de datos sin etiquetar. La principal novedad de nuestro método se basa en el aprendizaje por contraste. Demostramos cómo el aprendizaje por contraste se puede aplicar a la tarea de segmentación semántica y mostramos sus ventajas, especialmente cuando la disponibilidad de datos etiquetados es limitada logrando un nuevo estado del arte.

3. Nuevos *modelos de segmentación semántica de imágenes eficientes*. Desarrollamos modelos de segmentación semántica que son eficientes tanto en tiempo de ejecución, requisitos de memoria y requisitos de cálculo. Algunos de nuestros modelos pueden ejecutarse en CPU a altas velocidades con alta precisión. Esto es muy importante para configuraciones y aplicaciones reales, ya que las GPU de gama alta no siempre están disponibles.

4. Nuevos *métodos de segmentación semántica con sensores no RGB*. Proponemos un método para la segmentación de nubes de puntos LiDAR que combina operaciones de aprendizaje eficientes tanto en 2D como en 3D. Logra un rendimiento de segmentación excepcional a velocidades realmente rápidas. También mostramos cómo mejorar la robustez de estos modelos al abordar el problema de sobreajuste y adaptación de dominio. Además, mostramos el primer trabajo de segmentación semántica con cámaras de eventos, haciendo frente a la falta de datos etiquetados.

Estas contribuciones aportan avances significativos en el campo de la segmentación semántica para aplicaciones del mundo real. Para una mayor contribución a la comunidad científica, hemos liberado la implementación de todas las soluciones propuestas.

Summary

In computer vision, scene understanding aims at extracting useful information of a scene from raw sensor data. For instance, it can classify the whole image into a particular category (i.e. kitchen or living room) or identify important elements within it (i.e., bottles, cups on a table or surfaces). In this general context, semantic segmentation provides a semantic label to every single element of the raw data, e.g., to all image pixels or to all point cloud points. This information is essential for many applications relying on computer vision, such as AR, driving, medical or robotic applications. It provides computers with understanding about the environment needed to make autonomous decisions, or detailed information to people interacting with the intelligent systems.

The current state of the art for semantic segmentation is led by supervised deep learning methods. However, real-world scenarios and conditions introduce several challenges and restrictions for the application of these semantic segmentation models. This thesis tackles several of these challenges, namely, 1) the limited amount of labeled data available for training deep learning models, 2) the time and computation restrictions present in real time applications and/or in systems with limited computational power, such as a mobile phone or an IoT node, and 3) the ability to perform semantic segmentation when dealing with sensors other than the standard RGB camera.

The general contributions presented in this thesis are following:

1. A novel approach to address the problem of limited annotated data to train *semantic segmentation models from sparse annotations*. Fully supervised deep learning models are leading the state-of-the-art, but we show how to train them by only using a few sparsely labeled pixels in the training images. Our approach obtains similar performance than models trained with fully-labeled images. We demonstrate the relevance of this technique in environmental monitoring scenarios, where it is very common to have sparse image labels provided by human experts, as well as in more general domains.

2. Also dealing with limited training data, we propose a novel method for *semi-supervised semantic segmentation*, i.e., when there is only a small number of fully labeled images and a large set of unlabeled data. We demonstrate how contrastive learning can be applied to the semantic segmentation task and show its advantages, especially when the availability of labeled data is limited. Our approach improves state-of-the-art results, showing the potential of contrastive learning in this task. Learning from unlabeled data opens great opportunities for real-world scenarios since it is an economical solution.

3. Novel *efficient image semantic segmentation models*. We develop semantic segmentation models that are efficient both in execution time, memory requirements, and computation requirements. Some of our models are able to run in CPU at high speed rates with high accuracy. This is very important for real set-ups and applications since high-end GPUs are not always available. Building models that consume fewer resources, memory and time, would increase the range of applications that can benefit from them.

4. Novel *methods for semantic segmentation with non-RGB sensors*. We propose a novel method for LiDAR point cloud segmentation that combines efficient learning operations both in 2D and 3D. It surpasses state-of-the-art segmentation performance at really fast rates. We also show how to improve the robustness of these models tackling the overfitting and domain adaptation problem. Besides, we show the first work for semantic segmentation with event-based cameras, coping with the lack of labeled data.

To increase the impact of these contributions and ease their application in real-world settings, we have made available an open-source implementation of all proposed solutions to the scientific community.

Index

Index	iv
1 Introduction	1
1.1 Scene understanding	2
1.1.1 Semantic Segmentation	3
1.2 Challenges and Contributions	5
1.2.1 Semantic Segmentation with Limited Labeled Data	5
1.2.2 Efficient Semantic Segmentation	8
1.2.3 Semantic Segmentation with Other Sensors	9
1.3 Summary of Results	13
1.3.1 Publications	13
1.3.2 Code Released	14
1.4 Manuscript Organization	14
2 Semantic Segmentation from Few Labeled Pixels	15
2.1 Introduction	15
2.2 Related Work	18
2.2.1 Semantic Image Segmentation	18
2.2.2 Lack of Training Data	20
2.3 Training Dense Semantic Segmentation with Sparse Pixel Labels	21
2.3.1 Problem Formulation	21
2.3.2 Label Augmentation with Multi-Level Superpixels	22
2.3.3 Semantic Segmentation Architectures and Optimization	25
2.4 Datasets and Labels	27
2.4.1 Datasets	27
2.4.2 Reference Labels	28
2.5 Evaluation of the Proposed Approach	30
2.5.1 Labeling Augmentation with Multi-Level Superpixels	30
2.5.2 Analysis of Semantic Segmentation Methods	33
2.5.3 Training with Augmented Labels	34
2.6 A generic pre-trained coral encoder	36
2.6.1 Pre-Training and Fine-Tuning	36
2.6.2 Experiments	38
2.7 Applicability to Non-Coral Domains	40
2.7.1 Data and Evaluation	40
2.7.2 Approach Performance on Additional Domains	41
2.8 Conclusion	43

3	Semi-Supervised Semantic Segmentation	45
3.1	Introduction	45
3.2	Related Work	47
3.2.1	Contrastive Learning	47
3.2.2	Semi-Supervised Learning	47
3.2.3	Semi-Supervised Semantic Segmentation.	48
3.3	Method	48
3.3.1	Supervised Segmentation: \mathcal{L}_{sup}	49
3.3.2	Learning from Pseudo-labels: $\mathcal{L}_{\text{pseudo}}$	49
3.3.3	Direct Entropy Minimization: \mathcal{L}_{ent}	50
3.3.4	Contrastive Learning: $\mathcal{L}_{\text{contr}}$	50
3.4	Experiments	52
3.4.1	Datasets	52
3.4.2	Implementation details	52
3.4.3	Benchmark Experiments	54
3.4.4	Ablation Experiments	56
3.5	Conclusion	58
4	Efficient Semantic Segmentation	59
4.1	Efficient Semantic Segmentation to enhance V-SLAM Keyframe Selection	59
4.1.1	Related Work	61
4.1.2	Efficient semantic based keyframe selection on the robot CPU.	62
4.1.3	<i>MiniNet</i> network architecture	63
4.1.4	Experiments	65
4.1.5	Conclusion	70
4.2	MiniNet: An Efficient Semantic Segmentation ConvNet for Real-time Robotic Applications	71
4.2.1	Related work	72
4.2.2	Techniques for efficient semantic segmentation CNNs	73
4.2.3	Proposed Efficient semantic segmentation architecture	76
4.2.4	Evaluation	77
4.2.5	Analysis of MiniNet-v2 components	77
4.2.6	Benchmark evaluation	81
4.2.7	Conclusions	84
5	Semantic Segmentation With Other Sensors	85
5.1	3D-MiniNet: Fast and Efficient 3D LiDAR Semantic Segmentation	85
5.1.1	Related Work	87
5.1.2	3D-MiniNet: LiDAR point cloud segmentation	88
5.1.3	Experimental setup	92
5.1.4	Results	93
5.1.5	Conclusions	97
5.2	Domain Adaptation in LiDAR Semantic Segmentation	97
5.2.1	Related Work	99
5.2.2	Unsupervised Domain Adaptation For LiDAR Semantic Segmentation	100
5.2.3	Experimental Setup	102
5.2.4	Results	104
5.2.5	Conclusions	106

5.3	EV-SegNet: Semantic Segmentation for Event-based Cameras	107
5.3.1	Related work	108
5.3.2	From Events to Semantic Segmentation	109
5.3.3	Ev-Seg: Event-Segmentation Data	113
5.3.4	Experimental Validation	114
5.3.5	Conclusions	119
6	Conclusions	120
6.1	Semantic Segmentation with Limited Labeled Data	120
6.2	Efficient Semantic Segmentation	122
6.3	Semantic Segmentation with Other Sensors	122
6.4	Overall contribution.	124
6.5	Future Work	124

Chapter 1

Introduction

Computer vision is the field that studies how to extract information from visual data to provide machines with understanding of the surrounding environment, at various levels of detail and abstraction. To achieve this, computer vision addresses a wide range of different problems and tasks, such as object detection and tracking, semantic segmentation, 3D reconstruction, or localization. In recent years, deep learning has increased the performance of solutions for many of these computer vision tasks, boosting the interest of both research and industry communities, since applicability to real-world scenarios is becoming closer. Computer vision is, in fact, already present in our daily lives, facilitating many of our daily activities. It unlocks your mobile phone when you look at its camera or it improves the quality of our pictures based on the type of photo. It is at the core of all those cloud services processing the content of our photos and videos, it runs in our smart home appliances such as autonomous vacuum cleaners and it looks at the road and streets in driving assistance systems (see Figure 1.1). Besides, current research is pushing the application range further, achieving more and more real-world applications, including augmented reality applications [54], autonomous driving [19], AI-based health systems [237], or more applications to come.

This thesis studies the problem of scene understanding and, in particular, that of semantic segmentation of images. We next introduce these problems, discuss their current state of the art of this problems, with a focus on existing challenges and re-

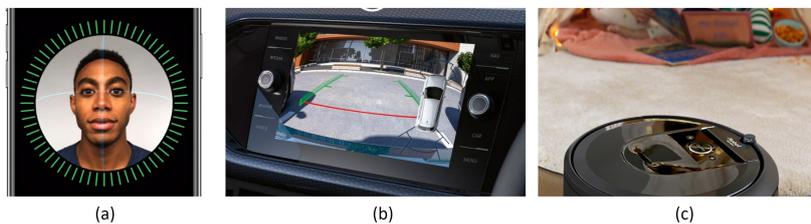


Figure 1.1: **Computer Vision in daily life.** Three examples of computer vision applications that have been introduced in our daily life: (a) computer vision in our mobile phones for face unlocking or camera applications; (b) driving assistance applications; (c) autonomous robots to help with daily chores such as vacuum cleaners. (Sources from <https://support.apple.com/>, <https://www.volkswagen.co.uk/> and <https://www.irobot.es/roomba> respectively)

strictions that come up when scene understanding methods are applied and deployed in real-world scenarios. We also present the main contributions of this thesis that tackle these challenges and provide a summary of the obtained results.

1.1 Scene understanding

One of the most ubiquitous computer vision task is scene understanding. Providing a semantic understanding of the scene captured by visual data to a computer allows the automation of many relevant tasks. It has applications in industry and manufacturing like product assembling or product flaws detection, in the health sector, e.g., to provide automatic medical reports, in robotics solving complex tasks for autonomous vacuum cleaners or autonomous driving. Depending on the goal or task, this semantic scene information can be provided at different levels (see Figure 1.2): scene classification (provides one tag for the image), object detection within the scene (provides one bounding box for each of the different objects in the image), scene semantic segmentation (provides a per-pixel semantic tag), or instance segmentation (provides a per-pixel semantic tag and a per-pixel object identifier).

Scene or image classification is a problem that has been studied for a long time, with works focusing on different image properties and hand-crafted description such as lu et al. [152]. In 2012, Krizhevsky et al. [130], presented a deep learning based approach that beat by a large margin previous methods and started to make neural networks great again. This work curved the path for the new deep learning era. We can see a summary of works highlighting deep learning based solutions in Nath et al. [181] and current state of the art solutions in [147].

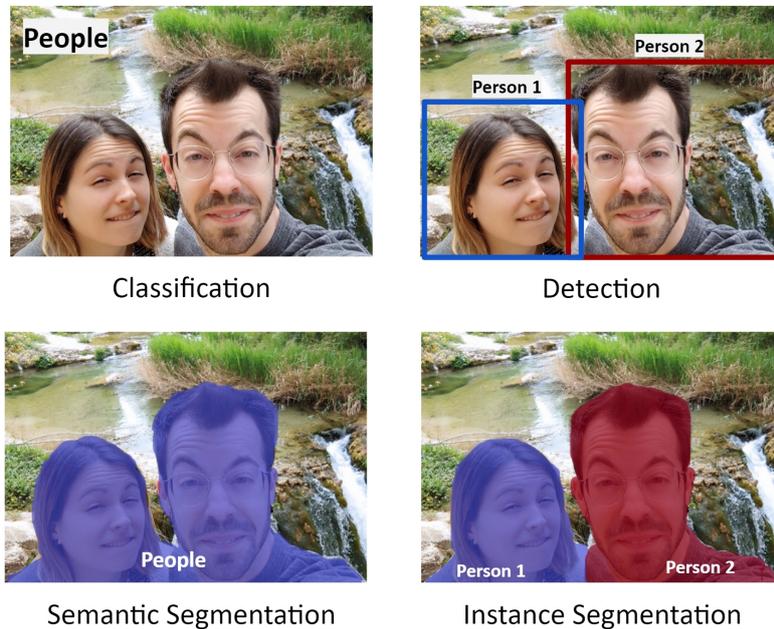


Figure 1.2: **Scene understanding.** Different scene understanding tasks that provide semantic information at different levels of detail.

Object detection and semantic segmentation tasks provide more semantic details of the scene than just a single label. While object detection provides bounding boxes of objects localizing them in the image, semantic segmentation assigns a semantic label to every pixel of an image (see this review for further information [73]). The combination of these two tasks is instance segmentation [170], providing both object-level and pixel-level information.

This thesis focuses on the problem of semantic segmentation, which is described and discussed in more detail next, including the evolution it had since the surge of deep learning.

1.1.1 Semantic Segmentation

As mentioned before, semantic segmentation provides a label at the pixel level. This type of detailed information can be exploited in many different ways. In Figure 1.3 we show some real-world applications of semantic segmentation such as medical applications or photography, or that can enable new applications, such as autonomous drone delivery, autonomous cars or virtual dressing. Indeed, the improvements of semantic segmentation are opening the door to improve real-world applications (e.g. medical imaging [205] or autonomous driving [16]) or even paving the way to new ones. In this context, semantic segmentation is facing one of those moments where it needs to run the last mile to boost its impact. As we will see below, this thesis will push the current state of the art closer to real-world applications, tackling several challenges that these scenarios introduce.

Related work. Similar to other scene understanding tasks, semantic segmentation have moved from commonly used hand-crafted features [68] to deep learning based

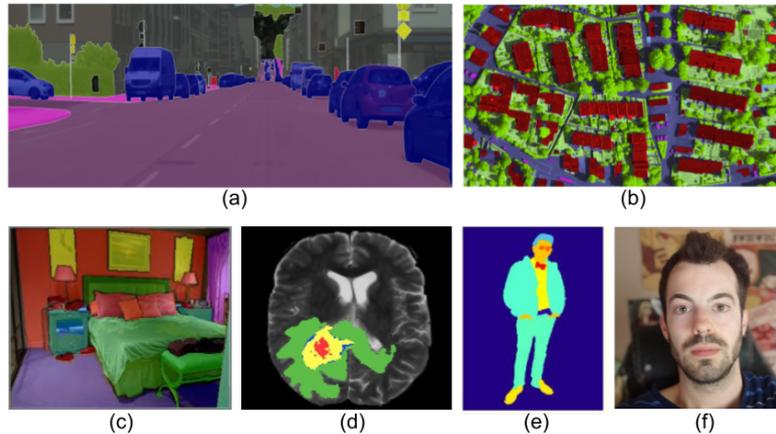


Figure 1.3: **Semantic segmentation.** Different semantic segmentation applications: (a)(b) semantic scene labeling for autonomous applications like cars or drones, (c) semantic labeling of all the components of indoor scenarios for indoors applications such as room virtual re-furnish, (d) semantic labeling for medical applications like X-ray cancer detection, (e) clothes applications like virtual dressing and, (f) background defocus as an example of a photography application. It can be appreciated how each pixel of the different images is colored, each color representing a semantic class.

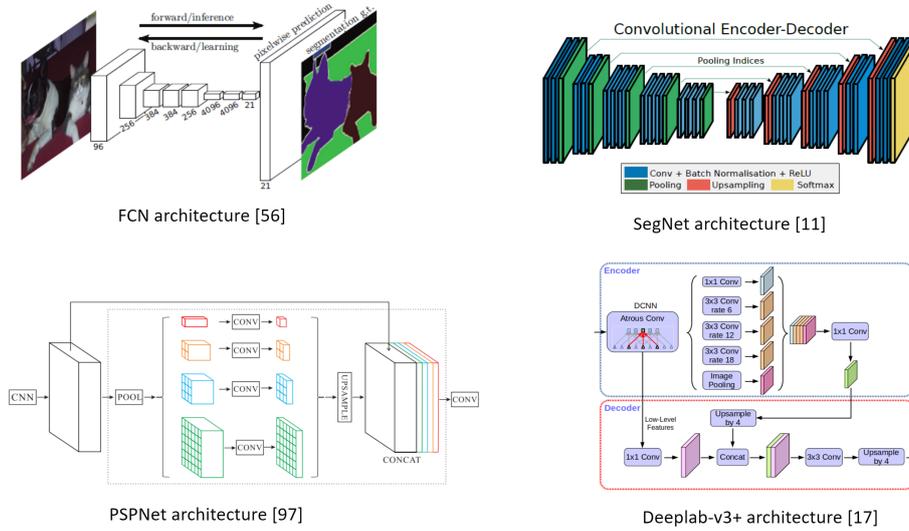


Figure 1.4: **Semantic segmentation architectures.** Comparison of different semantic segmentation architectures used in the literature.

techniques reaching impressive state-of-the-art results [73]. Fu et al. [68] presented a survey on different image segmentation methods before the deep learning era. Image segmentation methods were mostly based on colors and gradients of the image. Following [68], one can identify three categories depending on the different ways the segmentation is done: characteristic feature threshold or clustering, edge detection, and region extraction.

In recent years, semantic segmentation has received significant attention and has achieved great improvements, partially thanks to the performance boost that Convolutional Neural Networks (CNNs) have introduced in many vision tasks [89, 131, 150]. A survey on image segmentation, co-segmentation and object proposal by Zhu et al. [291] provides a detailed compilation of classical solutions for semantic segmentation, giving especial emphasis to superpixel methods. On the other hand, Garcia et al. [73] present a discussion of more recent deep learning-based approaches for semantic segmentation, covering from new architectures to common datasets. Current top performing methods for image semantic segmentation are based on deep learning [35, 36]. Semantic segmentation architectures are evolved from CNN architectures for classification tasks. These evolved architectures add a decoder module on top of the classification CNN to increase the learned feature representations resolution to achieve per pixel predictions. Therefore, the base classification architecture, typically named the encoder, learns features while reducing the resolution and the decoder upsamples the learned features and maps them into the segmentation result.

Fully Convolutional Neural Networks for Semantic Segmentation (FCNN) [150] are a seminal deep learning solution for this problem, carving the path for modern semantic segmentation architectures. The FCNN proposed to upsample the learned feature map of the classification CNNs using bilinear interpolation up to the input resolution. This way the authors could apply the typical classification loss, i.e., the cross-entropy loss [131], in a per-pixel fashion. Another early alternative proposed was the SegNet architecture [16], a work that proposes a symmetric encoder-decoder structure

using the unpooling operation as an upsampling layer. More recent deep learning-based approaches improve these earlier segmentation architectures by adding novel operations or modules proposed initially within CNNs architectures for classification tasks. For example, FC-DenseNet [112] follows DenseNet work [101] using dense modules. PSPNet [283] uses ResNet [91] as its encoder and introduces the Pyramid Pooling Module incorporated at the end of the CNN allowing to learn effective global contextual priors. One of the current top-performing methods is Deeplab-v3+ [36] whose encoder is based on Xception [45] using of depthwise separable convolutions [217] and dilated convolutions [265]. Figure 1.4 shows a visual summary of the evolution of the semantic segmentation architectures, comparing the different mentioned architectures.

The core of the current state of the art methods is based on a fully supervised approach that require large quantity of annotated data to achieve the reported high performance. This large amount of data annotations are not always available in all scenarios and not all applications and domains can afford the high labeling cost semantic segmentation requires. In parallel, the underlying deep models are getting more complex and demand high-end GPUs to be trained and run at acceptable frame-rates. Next section discusses in more detail the challenges semantic segmentation faces when brought to real-world applications and set-ups together with the contributions of this thesis to tackle these challenges.

1.2 Challenges and Contributions

The overall goal of this doctoral thesis is to advance the current state-of-the-art for semantic segmentation making emphasis on the requirements and needs from real-world applications. Our contributions address relevant challenges that top performing scene understanding methods, especially for semantic segmentation, face when they have to be used in real-world scenarios and set-ups. Each of the following subsections describes a different challenge and its relevance. For each one, we discuss relevant related work and present our proposed contributions to tackle each challenge.

1.2.1 Semantic Segmentation with Limited Labeled Data

Current state-of-the-art in semantic segmentation is led by convolutional neural network based methods [11, 37, 112]. These top-performing methods are supervised, i.e., they require a large set of annotated data to be able to generalize well [73, 291]. However, the availability of labeled data is a common bottleneck in a lot of real-world applications, especially for robotics or medical applications among others. Reducing the amount of labeled data needed for semantic segmentation is crucial since obtaining this per-pixel annotations is a an expensive and tedious task. In order to reduce this annotation cost, we can attempt to reach similar results with fewer annotation, or we can even reach better results by incorporating unlabeled data to the learning process.

Learning from limited labels is a challenging task. In this thesis we consider two main strategies for reducing the annotations when learning semantic segmentation models. The first strategy is weakly-supervised learning that learns from a weaker type of labels such as image-level labels, bounding boxes or sparse pixels. The second strategy is semi-supervised learning that assumes just a small subset of your data is labeled.

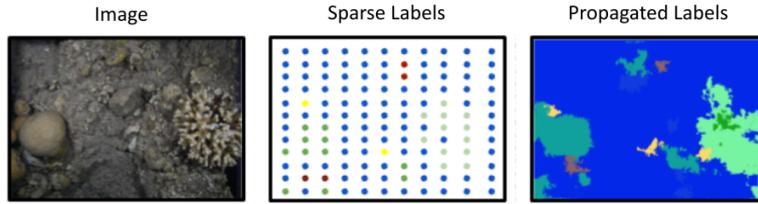


Figure 1.5: **Semantic segmentation from few labeled pixels.** Example of sparse labeled pixels and the propagated outcome of the proposed method.

Semantic Segmentation from Few Labeled Pixels: Weakly-supervised Learning

A common strategy for dealing with the lack of annotation is to learn from sparse or weakly labeled data. The term weak label refers to a different type of label one that provides less detail and therefore requires less annotation effort. For weakly-supervised semantic segmentation, a lot of different types of weak labels can be used, e.g., image labels [188, 192, 227], few labeled pixels (point-level) [18], scribbles [144, 230] or bounding boxes [52, 120]. The survey by Lu et al. [99] compares different methods to train semantic segmentation from weak labels. Methods for weakly supervised semantic segmentation depend on the type of weak labels. When image labels are available, the most common approach is to work with classification models and use the Class Activation Maps (CAMs) to detect the segmentation [2, 115, 214]. To get a segmentation, these approaches make use of the CAMs to extract the class that has a higher response on each pixel. When scribbles or only some pixels are available, the prevalent approach is to propagate the labels [13, 74] to get pseudo fully-labeled images. For the case of having bounding boxes as weak annotations, a segmentation model is learned directly, often using a dual-network (coarse-fine) approach [33, 156]. Here, the coarse model learns from the pixel annotations from bounding boxes. Then pseudo-labels are obtained using the coarse model to train the fine model.

Sparse labels for semantic segmentation are point-level or pixel-level annotations. This type of annotation consists of a set of few sparse annotated pixels for every image. Contrary to Bearman et al. [18], that learns directly from the few labeled pixels, this thesis studies how to get a fully-labeled image from sparse labeled pixels (see Figure 1.5). To do that, we propose a novel sparse labeling augmentation, i.e., propagation, method that enables training dense semantic segmentation models with sparse input labels, providing similar results to those obtained when training with densely labeled images, i.e., all pixels annotated. Therefore, we show how these propagated labels, acting as pseudo-labels are really effective in this type of scenarios where there are very few labeled pixels.

There are plenty of applications where it is common to find weak labels for semantic segmentation since semantic segmentation annotations are expensive. Besides, annotations require expert knowledge making the annotation cost expensive. The most common weak label for semantic segmentation is either image-level annotations or just some few labeled pixels. These annotations are really usual in monitoring applications like the case we study. Many different projects ranging from autonomous surveys of coral reef ecosystems [20, 159] to wildlife monitoring from aerial systems [94] focus on monitoring tasks and subsequent data analysis. To enable automatic processing of the data, semantic segmentation models for the different target domains are needed, but their use is often blocked or hampered due to the lack of dense labeling to train se-

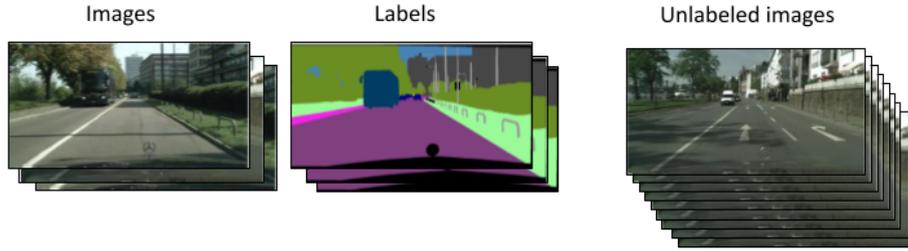


Figure 1.6: **Semi-supervised semantic segmentation.** A visual description of the semi-supervised semantic segmentation scenario where just a small set of the available data is labeled.

semantic segmentation models. As previously mentioned, this problem especially acute in domains where an expert is needed to label the images. To demonstrate the general applicability of the proposed strategy, we include experiments on different scenarios from other domains captured by different robotic platforms (aerial and urban scenarios). This contribution is explained in detail in Chapter 2.

The results of the work in this part of the thesis have been published in several conference [5, 7] and one journal [13] paper. Besides, the collaborations to apply part of these results in real-world ecological tasks, namely, monitoring of underwater have also lead to the publication of [273] and [274].

Semantic Segmentation from Partially Labeled Dataset: Semi-supervised Learning

Semi-supervised learning is another strategy to reduce the labeling cost, in this case by learning from only a few fully-labeled images and from an additional set of unlabeled images (see Figure 1.6). One common approach for this task is to make use of Generative Adversarial Networks (GANs) [80]. In [221] the authors proposed to generate new training samples by using a GAN, similar to the very recent work from Zhang et al. [278]. Differently, Hung et al. [107] proposed to train the discriminator to distinguish between confidence maps from labeled and unlabeled data predictions. Mittal et al. [172] made use of a two-branch approach, one branch enforcing low entropy predictions using a GAN approach and another branch for removing false-positive predictions using a Mean Teacher method [231]. Feng et al. [65] explore, in a recent work, a similar idea that introduces Dynamic Mutual Training (DMT). DMT uses two segmentation models and, the model’s disagreement is used to re-weight the loss. The DMT method also follows the multi-stage training protocol from CBC [64], where pseudo-labels are generated in an offline curriculum fashion. Other works used data augmentation methods, as a form of consistency regularization [67, 183], to make the network generalize better. These data augmentation techniques are a really popular solution to simulate having more labeled data, allowing your network to converge to a better solution.

In this thesis, we proposed a novel semi-supervised semantic segmentation method to reduce the amount of annotations required to train a semantic segmentation model. The idea, as mentioned above, is to use the small subset of the available data that is labeled, while also extracting knowledge from unlabeled samples. We present a novel approach for semi-supervised semantic segmentation based on a novel representation learning module. This module, based on contrastive learning, enforces the segmentation network to yield similar pixel-level feature representations for same-class samples

across the whole dataset. To achieve this, we maintain a memory bank continuously updated with feature vectors from labeled data and, in an end-to-end training, the features from both labeled and unlabeled data are optimized to be similar to same-class samples from the memory bank. We show that our approach outperforms the current state-of-the-art for semi-supervised semantic segmentation and semi-supervised domain adaptation on well-known public benchmarks, with larger improvements on the most challenging scenarios, i.e., less available labeled data [12]. This proposed per-pixel contrastive learning approach can be potentially applied to other segmentation tasks, like instance or panoptic segmentation, or even to the object detection problem.

1.2.2 Efficient Semantic Segmentation

Efficiency is another important factor to consider when aiming to apply semantic segmentation solutions in real-world settings. Semantic segmentation is an essential task for scene understanding but it is unfeasible to run many of the top-performing models on CPUs, or even at low-powered GPUs, at the high frame-rates required in many real applications, such as autonomous driving systems. Figure 1.7 shows three examples of autonomous robots that require fast and efficient scene understanding to execute their tasks in a safely and proper manner fashion.

CNN models, especially for semantic segmentation, usually require high-end GPUs to run inferences in near real-time. Unfortunately, the availability of this type of GPUs in many applications like mobile phones or robotic platforms is not common, partially due to their high cost or physical restrictions such as space or weight. Even autonomous vehicles cannot afford to have one or several high end GPUs for all the possible tasks that can be run with deep learning methods. Therefore, reducing the computational cost of semantic segmentation models would allow to increase the platforms and applications that can use and deploy these segmentation models.

In the last years, we have seen an increasing interest in deep learning solutions for real-time applications on low-power GPUs or even for CPUs. Lately, many works have focused on reducing CNNs memory and computational requirements, which directly affects energy consumption and execution time. There are different ways this problem has been tackled. Some approaches focus their contributions on how to transfer the knowledge from a large and accurate model to a small efficient one, i.e., knowledge distillation [93, 193]. Other works focus their attention on carefully choosing the parameter data types. Quantized models [103] or binary networks [51] are recent



Figure 1.7: **Efficient semantic segmentation.** Small autonomous robotic platforms are examples with efficiency requirements to be able to run scene understanding models onboard. These robots require a semantic understanding of their environments to be able to act or interact with it and to move in a safe fashion. (Sources from <https://amazon.com/Amazon-Prime-Air>, <https://www.amazon.jobs/es/teams/amazon-scout> and <https://www.bostondynamics.com/spot> respectively)

works studying the effect of using less precise data types. Increased efficiency can also be achieved through post-processing methods such as pruning [174], i.e., reducing a trained CNN without losing accuracy. Other works study novel network operations such as depth-wise separable convolutions [217]. Finally, other works propose novel CNN architectures directly focused on the target task. In particular, for efficient segmentation we find architectures such as Enet [187] or ERFNet [202]. These works propose different modules and strategies to perform semantic segmentation more efficiently.

In this part of the thesis, we study how to improve existing work on efficient architectures for semantic segmentation, especially, through efficient operations. We also show how these efficient architectures can be used both for CPU and low-powered GPUs to help and support real-world applications. In our work, efficiency is taken into account in several ways: speed, memory requirements and, computation (floating point operations). Our main contribution in this regard is the novel architecture of MiniNet. First presented in [10] and refined in a second version, MiniNet v2, published later in [11].

MiniNet is a fast and efficient semantic segmentation network that can run in CPU in real-time. It provides sufficient accuracy and is significantly faster and more efficient than related works, providing useful semantic information to real-world applications. As a real-world application, we integrate this in a robot for the application of selected keyframes in the Visual-SLAM (V-SLAM) pipeline. We propose a novel keyframe selection strategy based on image quality and semantic information by using our MiniNet model. While commonly used V-SLAM methods select keyframes based only on relative displacements and amount of tracked feature points, our strategy to select more carefully these keyframes allows the robotic systems to make better use of them. With minimal computational cost, we show that our selection includes more relevant keyframes, which are useful for additional posterior recognition tasks, without penalizing the existing ones, mainly place recognition. We demonstrate our hypothesis with several public datasets with challenging robotic data.

MiniNet-v2 [11] improves the seminal architecture. The improvement is mainly based on two contributions. The first one is the proposed multi-dilation depthwise separable convolution which replaces the standard separable convolutions of MiniNet. The second contribution is an analysis on some common efficient strategies for semantic segmentation is performed to improve the efficiency of the proposed architecture. These strategies can be applied to any top-performing segmentation model allowing to reduce their computation and memory requirements and boosting their frame-rates, making segmentation models more accessible for real-world applications.

We validate and analyze the details of both MiniNet versions through a comprehensive set of experiments on public benchmarks (Cityscapes, Camvid, and COCO-Text datasets), showing their benefits over relevant prior work.

1.2.3 Semantic Segmentation with Other Sensors

The most common sensor for vision applications is the RGB camera. This sensor is affordable and provides rich visual information to perform almost any vision task, like scene understanding. RGB cameras provide a 2D representation of a subset of their surrounding environment. So far, all the proposed solutions in this thesis for semantic segmentation challenges, have been focused on RGB cameras. However, in many scenarios, especially for autonomous systems and robotic applications, other sensors can provide additional or alternative useful information. This last part of the thesis is

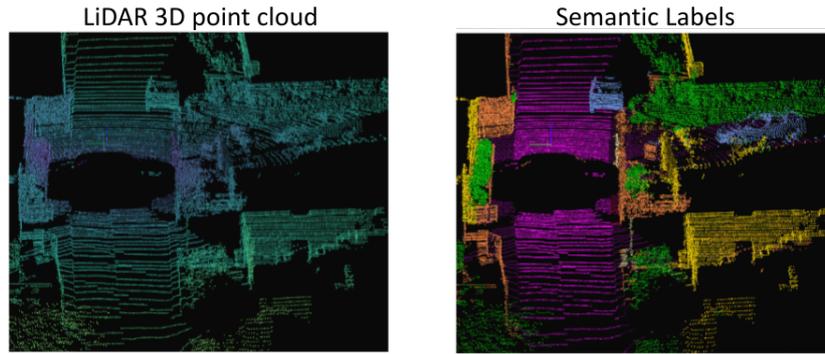


Figure 1.8: **3D LiDAR semantic segmentation.** Example of data from a LiDAR sensor. 3D point cloud extracted from LiDAR (colors of LiDAR represent the distance to the capturing sensor) and its corresponding semantic labels (blue color for cars, green for vegetation, purple for road, orange for walls, and yellow for building).

focused on the use of two sensors of high interest due to the complementary data they can provide with respect to standard RGB vision sensors for vision applications like autonomous robots: LiDAR sensor and event-based cameras.

Semantic Segmentation for LiDAR. The LiDAR sensor provides a 3D sparse representation of the surrounded scene, i.e., a 3D point cloud (see Figure 1.8). While in RGB cameras the color and gradient information are the most distinctive features to distinguish the different elements of the scene, in 3D point clouds, the 3D structure and the spatial relationship of the 3D points are the main characteristics to analyze. The semantic segmentation of LiDAR data provides very useful information to autonomous robots when performing tasks such as Simultaneous Localization And Mapping (SLAM) [113, 284], autonomous driving [169] or inventory tasks [41]. Semantic segmentation provides key information such as drivable areas or identifying possible dynamic objects. In this type of scenarios, as discussed also in previous subsection 1.2.2, it is critical to have models that provide accurate semantic information in a fast and efficient manner. This is particularly challenging working with 3D LiDAR data because point clouds are an unstructured and unordered set of points contrary to 2D images. Besides, the size of LiDAR point cloud tend to be large. There are different existing strategies to tackle LiDAR semantic segmentation. On one hand, the commonly called *point-based approaches* [190, 191, 232] tackle this problem directly executing 3D point-based operations, which is computationally expensive to operate at high frame-rates. This inefficiency is especially highlighted with LiDAR data since LiDAR point clouds tend to contain a large set of points. Other approaches follow an intermediary step of building 3D representations like 3D voxels [150] that although converting the point cloud into a structured representation, this strategy still require expensive operations like 3D convolutions. Recent results on fast [169] and parameter-efficient [281] semantic segmentation models are facilitating the adoption of semantic segmentation in real-world robotic applications [19, 140]. These works project the 3D information into a 2D image (*projection-based approaches*) in order to work with 2D CNNs that are more efficient [58, 169, 251, 256, 257]. This last part of the thesis addresses the LiDAR semantic segmentation problem by building an novel efficient network for LiDAR point clouds, combining both 3D operations with 2D convolutions: 3D-MiniNet [9].

3D-MiniNet first learns a 2D representation from the raw points through a novel projection that extracts local and global information from the 3D data. This representation is fed to an efficient 2D Fully Convolutional Neural Network (FCNN) that produces a 2D semantic segmentation. These 2D semantic labels are re-projected back to the 3D space and enhanced through a post-processing module. We validate our approach on well-known public benchmarks, where 3D-MiniNet gets state-of-the-art results while being faster and more parameter-efficient than previous methods.

Although 3D-MiniNet gets high performances on these benchmarks, we realized it was not generalizing when applied to other domains. This problem, i.e., the **domain adaptation problem**, is a common issue in real-world applications. This appears when your model overfits to your training data, i.e., when your method can perform accurately on your training data but it performs badly on the test data or real unseen data. This is due to the difference in the data distributions of the training and test data. This is a common issue in Machine Learning. Therefore, although we can build an accurate and efficient model for 3D LiDAR semantic segmentation, it is very difficult to make it work on very different real scenarios. There are existing techniques that aim to eliminate this effect. Existing works for domain adaptation in semantic segmentation focus on RGB data [38, 142, 247, 293]. Most of them try to minimize the distribution shift between two different domains. Most works follow the idea that the input data or features from two different domains should be indistinguishable. The common way to force this idea into the deep learning pipeline is to perform adversarial training. This approach has shown good performance at pixel space [266], at feature space [96] and at output space [247]. Other works tackle this by relying on a loss function that minimizes the entropy of the unseen unlabeled domain output probabilities.

In this last part of the thesis, we tackle this domain adaptation problem for 3D LiDAR semantic segmentation. We propose simple but effective strategies to reduce the domain shift by aligning the data distribution on the input space. Besides, we present a learning-based module to align the distribution of the semantic classes of the target domain to the source domain. Our approach achieves new state-of-the-art results on three different public datasets, which showcase adaptation to three different domains. [6]

Semantic segmentation for event-based cameras. Event cameras [69] are a promising sensor able to register intensity changes of the captured environment. In contrast to conventional cameras, this sensor does not acquire images at a fixed frame-rate. These cameras, as their name suggests, capture events and record a stream of asynchronous events. An event indicates an intensity change at a specific moment and at a particular pixel. Event cameras offer multiple advantages over more conventional cameras such as the high temporal resolution (capturing frequency) which allows the capture of multiple events in microseconds or its very high dynamic range, which allows the information capture at difficult lighting environments. Several recent works [69], have shown their benefits in some visual recognition tasks, such as classification [160] or object detection [171], emphasizing that event cameras are natural motion detectors. However, the problem of semantic segmentation had been barely studied before the work developed during this thesis. This lack of results on semantic segmentation is due several reasons. The main cause is the lack of real labeled data for this sensor, which lately it has been mitigated by using virtual simulators [196]. Contrary to RGB images, where humans can recognize the elements within the image, event-based data is not that easy to extract information from, since the visualization of this data is less

understandable for humans. Another reason for the lack of semantic segmentation works for this sensor is that, at the time we worked on it, it was still a relatively new and recent sensor and there were still more basic research challenges such as finding a proper representation for this data.

The last part of this thesis studies semantic segmentation for event-based cameras. The main contribution lies on how to train semantic segmentation for event-based cameras when there is a lack of annotations. The proposed method uses a synchronized grayscale camera with the event-based camera. Instead of annotating the grayscale images, a semantic segmentation model trained on a similar domain is used to create pseudo-labels with no human involvement for the annotation. This was at the time the first work to propose a learning method to learn semantic segmentation from these type of event data [8].

1.3 Summary of Results

1.3.1 Publications

The results of the work developed during this PhD thesis have been published in several top international conferences and journals:

Semantic Segmentation from Few Labeled Pixels: Weakly-supervised Learning

- I. Alonso, A. Cambra, A. Muñoz, T. Treibitz, A. C. Murillo. **Coral-Segmentation: Training Dense Labeling Models with Sparse Ground Truth**. *ICCV workshop on Visual Wildlife Monitoring, ICCVW 2017*.
- I. Alonso, A. C. Murillo. **Semantic Segmentation from Sparse Labeling using Multi-Level Superpixels**. *International Conference on Intelligent Robots and Systems, IROS 2018*.
- I. Alonso, M. Yuval, G. Eyal, T. Treibitz, A. C. Murillo. **CoralSeg: Learning Coral Segmentation from Sparse Annotations**. *Journal of Field Robotics, 2019*.
- M. Yuval, I. Alonso, G. Eyal, D. Tchernov, Y. Loya, A. C. Murillo, T. Treibitz **Repeatable Semantic Reef-Mapping Through Photogrammetry and Label-Augmentation**. *Remote Sensing, 2021*.

Semantic Segmentation from Partially Labeled Dataset: Semi-supervised Learning.

- I. Alonso, A. Sabater, D. Ferstl, L. Montesano, A. C. Murillo. **Semi-Supervised Semantic Segmentation with Pixel-Level Contrastive Learning from a class-wise Memory Bank**. *Under review*.

Efficient Semantic Segmentation

- I. Alonso, L. Riazuelo, A. C. Murillo. **Enhancing V-SLAM Keyframe Selection with an Efficient ConvNet for Semantic Analysis**. *International Conference on Robotics and Automation, ICRA 2019*.
- I. Alonso, L. Riazuelo, A. C. Murillo. **MiniNet: An Efficient Semantic Segmentation ConvNet for Real-time Robotic Applications**. *IEEE Transactions on Robotics (T-RO), 2020*.

Semantic Segmentation with Other Sensors

- I. Alonso, L. Montesano, A. C. Murillo. **3D-MiniNet: Learning a 2D Representation from Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation**. *IEEE Robotics and Automation Letters (RA-L), 2020*. Presented on *International Conference on Intelligent Robots and Systems, IROS 2020*.
- I. Alonso, L. Riazuelo, L. Montesano, A. C. Murillo **Domain Adaptation in LiDAR Semantic Segmentation**. *International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2021*.

- I. Alonso, A. C. Murillo. **EV-SegNet: Semantic Segmentation for Event-based Cameras**. *Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2019*.

1.3.2 Code Released

During the development of this thesis, we have released the following code repositories, some including also new data released.

Semantic Segmentation from Sparse Labeled Pixels. We make the developed tools, code, models, and data publicly available at <https://sites.google.com/a/unizar.es/semanticseg/home>.

Semi-Supervised Semantic Segmentation. Semi-supervised semantic segmentation approach will be made publicly available at <https://github.com/Shathe/SemiSeg-Contrastive>.

Efficient Semantic Segmentation. The code of the two MiniNet versions is available at <https://github.com/Shathe/MiniNet-v2>.

Semantic Segmentation with Other Sensors. Code for the LiDAR semantic segmentation is available at <https://github.com/Shathe/3D-MiniNet>. Code for event-based cameras, at <https://github.com/Shathe/Ev-SegNet>.

1.4 Manuscript Organization

The following chapters describe in detail the contributions presented above: semantic segmentation from sparse labeled pixels (Chapter 2), semantic segmentation with limited data (Chapter 3), efficient semantic segmentation (Chapter 4), semantic segmentation with other sensors (Chapter 5). Each of these chapters is self-contained, including a brief specific related work section, and the corresponding conclusions and discussions. Chapter 6 outlines the general conclusions of the thesis, as well as our vision of future directions.

Chapter 2

Semantic Segmentation from Few Labeled Pixels

Current state-of-the-art for semantic segmentation follows supervised strategies based on deep learning that usually require a lot of labeled data to train the segmentation model. However, this large amount of labeled data is not always available. This chapter of the thesis tackles the problem of limited labeled data availability for semantic segmentation. This is one of the common challenges faced by many of the best performing methods for semantic segmentation, based on supervised deep learning, when applied in real-world scenarios. As mentioned in previous introductory chapter, the first contribution of this thesis is focused on learning semantic segmentation models from sparse labels, i.e., few labeled pixels for each image. Having the ability to learn semantic segmentation models from just a few labeled pixels will enable applications with low annotation budget to successfully apply semantic segmentation tasks. This is the case, for example, of many environmental monitoring applications, where an expert is needed to label the images. In particular, this contribution has been developed with especial attention to the use case of coral reef monitoring applications, since in this field is common to find image datasets with just a few sparse points labeled by experts. We demonstrate how to successfully train coral reef semantic segmentation models from just a few labeled pixels per image.

2.1 Introduction

Advances in robotics have facilitated the acquisition of data in challenging environments, such as underwater [30, 79] and aerial [127] surveys. In particular, visual sensors are a widely used tool that requires little expertise to produce massive datasets. Effectively, researchers are able to rapidly document large areas with high-resolution images, shifting the bottleneck in wide-scale ecological research and monitoring towards image analysis over image acquisition. When done manually, the extraction of useful data from these collections is an onerous task, which urgently demands new solutions and automation.

Semantic image segmentation is the task of automatically providing a complete understanding of scenes captured in images. The impressive development of deep neural networks, especially convolutional neural networks (CNNs) [73], has led to a significant improvement in semantic segmentation approaches in recent years. Many robotic

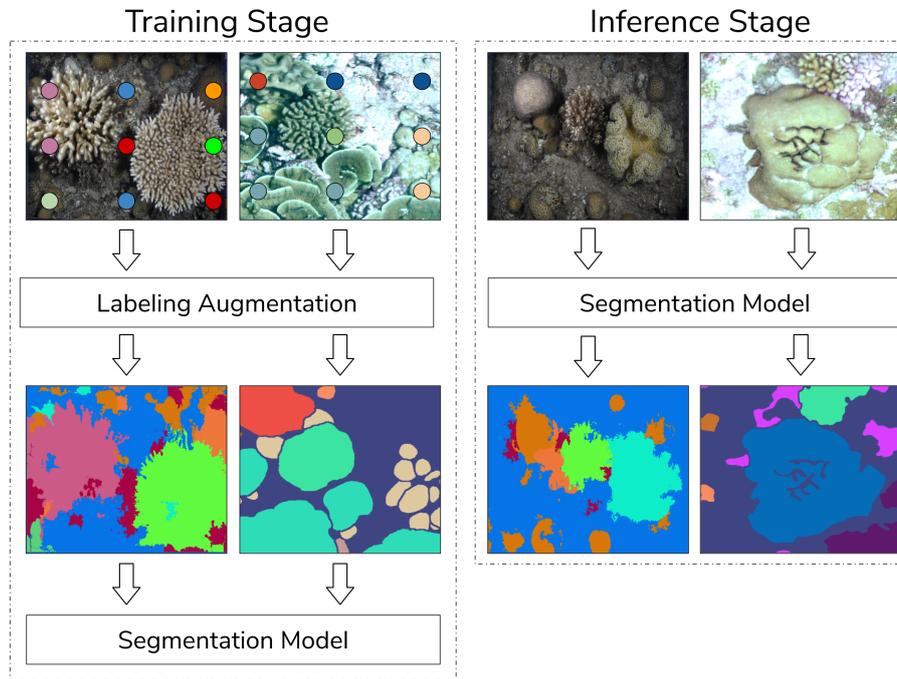


Figure 2.1: **Training semantic segmentation models from sparse labels.** In the training stage, we demonstrate how to augment the sparse labels into fully labeled (dense) images, which are used to train the semantic segmentation model. This model is used in later inference stages to obtain dense segmentation of new input images without any supervision. Our pipeline, requiring a much lower labeling effort than prior work, enables effective training of semantic segmentation models.

applications benefited from these improvements, e.g., autonomous driving [154] and object detection and manipulation [255]. For training, however, these methods require extensive amounts of pixel-level labeled data. Dense pixel-level annotation is time-consuming and often requires specific expertise, making the labeling process highly expensive and limited in domains that could benefit from it significantly, such as survey tasks [21, 244]. For instance, there is abundant underwater monitoring data in the CoralNet¹ project [20], from many different locations, labeled by marine biology experts. Yet, each image is only sparsely labeled, having on average 50-200 labeled pixels. Here we suggest a novel approach of *learning dense labeling from sparse labels* (Fig. 2.1). It enables the application of recent developments in deep learning for semantic segmentation in a wider range of domains including coral segmentation demonstrated here. Many other monitoring applications, such as traffic or agricultural monitoring [168] will also be able to reap the benefits of this work.

Coral monitoring challenges. The oceanic underwater environment has remained severely overlooked despite the fact that the ocean covers 71% of the world's surface [246]. Coral reefs are among the most important marine habitats, occupying an important portion of the ocean, and hosting a substantial amount of all known marine

¹<https://coralnet.ucsd.edu/>

species [194]. Most reef-building corals are colonial organisms of the phylum Cnidaria. Their growth creates epic structures that can be seen from space. These structures not only harbor some of the world’s most diverse ecosystems, but also provide valuable services and goods such as shoreline protection, habitat maintenance, seafood products, recreation, and tourism. Furthermore, due to their immobility, corals have developed an arsenal of chemical substances that hold great medicinal potential [32, 95, 173, 194].

Today, coral reefs face severe threats as a result of climate change and anthropogenic-related stress [105]. Ocean acidification, rising sea surface temperature, over-fishing, eutrophication, sedimentation [63] and pollution [15, 40, 95, 201] are only a few examples of these menaces. Coral reef ecosystems have suffered massive declines over the past decades, resulting in a marine environmental crisis [104, 106].

In coral reefs, dynamics occur over many spatial scales that range from millimeters to kilometers, and the zonation and growth of dominant species form salient patterns [81, 108]. Studying the complex biological systems together with the structures modified by coral growth and decay remains a challenge in coral reef studies [222]. In fact, this hurdle stresses the need for a cross-scale, highly automated approach.

The specific challenges in coral recognition from benthic images are linked directly to the difficulties in underwater imaging and the adaptable nature of corals expressed in their exceptional phenotypic plasticity. Underwater images suffer from color distortion and low contrast. The color of an object imaged underwater varies with distance and the water’s optical properties, depending on depth and water type. These dependencies are wavelength-specific, making color in underwater images an unstable source of information [3, 22], unless corrected [4]. Scleractinian corals are known to display morphological plasticity, i.e., intra-specific variations in the shape and form of colonial units [235]. These variations represent the feedback between the organism’s developmental plan and the surrounding ecological context and settings [211]. They are governed by biotic and abiotic factors such as interspecific interactions, and light regimes along a depth gradient [62]—all of which make automated image labeling difficult. Moreover, the overall community structure of coral reef assemblages varies greatly, spatially and temporally [49, 92]. Depth-related zonation, a predominant characteristic of coral reefs [108], also adds to the challenge. Such dissimilarities must be taken into consideration in benthic image analysis, and highlight the need for an adaptive identification tool that is robust to different underwater scenes and can be utilized across an assortment of datasets.

To address this shortcoming, several tools were developed for the annotation of marine images and videos [78]. Although some of these offer point predictions [20] and area measurements [128], to the best of our knowledge, none possess the novel capabilities of our suggested framework: to learn semantic segmentation from sparse annotations through adaptive labeling augmentation.

Automated semantic segmentation represents a leap-forward in benthic image analysis as it not only provides partial presence/absent data but also allows measurement of morphological attributes such as size-frequency distribution of key groups across an image set and observation of wide scale patterns with minimal labeling effort. As underwater images present one of the hardest use cases for image analysis, our methodology can be adapted easily to a terrestrial setting such as drone-image analysis.

Our experimental results demonstrate that the proposed augmentation of sparse labeling, despite being less accurate than manual annotation, provides valuable and effective information to train a state-of-the-art segmentation model. The results are comparable to approaches trained on densely labeled images, while having the advantage of less intensive annotation requirements. The results also show how different

losses for semantic segmentation and architectures affect the different important metrics for semantic segmentation. The presented encoder trained on CoralNet data (half a million images) enhances the semantic segmentation models when fine-tuning the training. This is a similar concept to that of ImageNet [56] but specific to coral reef images. We show how this encoder helps semantic segmentation models learn more general and better features for coral images. Finally, the experimental results show that our method can be applied to and provide the same benefits for other domains, increasing the number of robotic applications that can benefit from it.

The specific **contributions** of this work are:

- A *novel sparse label augmentation method* that enables training dense semantic segmentation models with sparse input labels, providing similar results to those obtained when training with dense labels. This is particularly significant for many ecological applications since most expert labeling efforts consist of sparse labels, and manual dense labeling of many images is essentially infeasible. To demonstrate the general applicability of the proposed strategy, we include experiments on different datasets from other domains captured by different robotic platforms (aerial and urban scenarios).
- We train and release a *generic encoder* for *coral imagery*, trained on over half a million coral reef images. Our experiments demonstrate that this model has learned generic representations for coral imagery that help learning segmentation models for new specific scenarios with few labeled samples available.
- We also make available the new data and developed tools.²
- A comparison of different well-known deep learning architectures for semantic segmentation applied to underwater coral reef imagery. We cover not only architectures but also common loss functions and propose a new, more suitable variation of the cross-entropy loss for this problem.

2.2 Related Work

This section discusses work from areas most relevant to ours: methods for and state-of-the-art of semantic segmentation with special attention on underwater imagery segmentation and strategies to deal with a lack of the required training data, i.e., sparse or weak labels.

2.2.1 Semantic Image Segmentation

Semantic segmentation is a visual recognition problem consisting of assigning a semantic label to each pixel in the image. The state-of-the-art in this task is currently achieved by solutions based on deep learning, most of them proposing different variations of fully convolutional networks (CNNs) [35, 36, 112, 150]. Some existing solutions for semantic segmentation target instance-level semantic segmentation, e.g., Mask-RCNN [89], which includes three main steps: region proposal, binary segmentation, and classification. Other solutions, such as DeepLabv3+ [36], target class-level

²Tools, model and data publicly available on <https://sites.google.com/a/unizar.es/semanticseg/home>.

semantic segmentation. DeepLabv3+ is a top performing CNN for semantic segmentation and the base architecture of our work.

Prior to the surge of deep learning approaches, several algorithms based on superpixel segmentation techniques [223] were used for this task. These approaches cluster image pixels into several groups of similar and connected pixels (i.e., superpixels). Such approaches classify the superpixels or a superpixel-based labeling propagation [167, 234]. The survey by [291] of image segmentation provides a detailed compilation of more conventional solutions for semantic segmentation. A later survey [73] presents a discussion of more recent deep learning-based approaches for semantic segmentation, ranging from new architectures to common datasets. Our work exploits both types of approaches. As we discuss later, while the CNN-based models are the core of our segmentation process, we show that the superpixels are very effective in augmenting sparse labels.

Coral reef community structure analysis. Community ecology is the field that studies the interactions of species that co-occur in space and time [177]. Diversity, a broad term that describes the numerical composition of species, is a feature of ecological communities [210]. Here, we focus on coral reef communities; the Macro-benthos, and more specifically, Scleractinian corals.

Traditionally, classification, mapping, and depiction of coral reef community structure has been performed *in situ* by scuba divers trained in marine ecology. Common methods for systematic depiction in quantitative studies of the reef substrate use quadrats and line transects as references to estimate attributes such as live cover, species richness, biodiversity, and population density [135, 151, 222, 253]. These methods are borrowed from terrestrial ecology, where they are simple to conduct. When studying the reef and its inhabitants *in situ*, however, divers face limitations such as depth and time. In addition, community structure classification is prone to human bias. Technological developments and engineering have helped to surmount these challenges using an array of sensors—mainly visual and acoustic. Image collections of the substrate present a repeatable, minimal impact tool for observation-based studies. Scalable approaches such as photo-mosaics now allow scientists to capture and systematically describe large-scale ecological phenomena with genus-specific resolution [66, 79, 155, 218]. Previous work [20] investigated automated approaches for determining the spatial distribution of the various organisms in a coral reef ecosystem using survey images. In particular, this work cropped image patches around the sparse labels and then performed image classification using support vector machine methods. Other works performed coral reef analysis using machine learning methods such as k-nearest neighbors [159, 161, 215]. Nevertheless, as previously mentioned, deep learning approaches are achieving state-of-the-art performance in classification, detection and segmentation tasks [73] including coral reefs analysis [175]. Deep learning approaches have also been shown to perform better when learning from multimodal data. For example, [21, 294] have presented a wide field-of-view fluorescence imaging system called FluorIS, which classifies coral species better than when only using RGB images.

More recent approaches are shifting to semantic segmentation, which is able to give more detailed information (pixel-level) than only classification. The first approaches performed image patch classification to thereafter reconstruct the segmentation of the entire image [159, 215]. These kinds of patch-based approaches, however, typically have low accuracy near the edges of the segmented regions. To get the fully segmented

image, moreover, they also need to be executed the same amount of times as the number of patches cropped from the image.

In contrast, our work presents an approach to directly learn semantic segmentation models from sparse ground truth labels, as demonstrated later, achieving better performance than earlier works based on patches. This approach is based on our earlier works [5, 7], which exploit superpixel segmentation to propagate the training labels, as we detail in Section 2.2.2. Another recent work, demonstrating the benefits of incorporating the use of superpixels for semantic segmentation tasks using CNNs [122], used superpixel segmentation to build a tool to facilitate the labeling process.

2.2.2 Lack of Training Data

As previously mentioned, many different projects ranging from autonomous surveys of coral reef ecosystems [20, 159] to wildlife monitoring from aerial systems [94] focus on monitoring tasks and subsequent data analysis. To enable automatic processing of the data, semantic segmentation models for the different target domains are needed, but their use is often blocked or hampered due to the lack of dense labeling to train semantic segmentation models, especially in domains where an expert is needed to label the images. This common situation motivates the solution presented here: our method to surmount the lack of labeled training data. Before presenting our proposed methodology, we review several methods for overcoming this problem found in prior work.

Models for weakly labeled data. A common strategy for dealing with the lack of annotation is to build approaches that are able to learn from sparse or weakly labeled data. The survey by [99] compares different methods to train semantic segmentation from noisy and weak labels. The work discusses these problems in detail and presents some solutions.

Several recent approaches show how to make use of *per image labels* to obtain per pixel image segmentation models. This work [129] proposes a new composite loss function to train fully convolutional networks directly from image-level labels. Another study [59] proposes a two-step approach: first, teach a CNN classification model trained on image-level labels to learn good representations and, then, use the learned feature maps to get the segmentation result.

Notwithstanding, several recent works have studied learning from *sparse labels* from different perspectives. A recent work [238] proposes a new CNN architecture, Sparsity Invariant CNN, focused on reconstructing a dense depth map from sparse LiDAR information. This approach outputs continuous values in contrast to the classification labels. The authors work with sparse convolutions to learn directly from sparse labeling, and show successful results with levels of sparsity between 5% and 70%. Label propagation was also used in [245], who show how to simultaneously learn a label-propagator and the image segmentation model, both with deep learning architectures. This approach propagates the ground truth labels from a few traces to estimate the main object boundaries in the image and provides a label for each pixel. In contrast, we use superpixel-based method for the propagation, resulting in better results.

Generating new data. Another strategy for dealing with the lack of training data is to generate additional or new data similar to the real data. Generating data by modifying its original form is a fairly common solution. Many works have used variations of

this method, including the well-known Alexnet model [131], which was trained using image augmentation by applying image flips and translations and altering RGB values. A more recent data augmentation solution is to generate synthetic data [84, 206]. This strategy provides perfect ground truth labels through image rendering. These types of methods do not always transfer generated data to real data properly, in part because, in many situations, it is hard to simulate the right amount of variability needed for the training data. Another recent work [225] describes how to adapt an existing model when there is no training data available for the new domain.

Contrary to the above-mentioned approaches, we study an alternative but complementary path that combines the idea of data generation (augmenting the sparse labels) and CNN models for segmentation. We demonstrate how to augment the sparse labeling using superpixel segmentation algorithms and study the effects.

This work is not the first one that uses superpixel segmentation to enhance annotation pipelines. Preliminary results of training dense segmentation models with augmented sparse labels were shown in our earlier work [5, 7]. Other works have also built annotation tools using this approach. For example, [254] proposes a superpixel labeling interface for semantic image annotation. Very similar to [254], Labelbox³, an online platform for semantic image annotation, commercialized this idea. In contrast to these annotation tools, the present work introduces an iterative (multi-level) and automatic method for augmenting sparse labels. Thanks to this iterative approach, the annotator does not need to change parameters such as the superpixels sizes or the number of generated superpixels. Instead, we iteratively build several levels of superpixels that perform this task automatically.

The single-level strategy that uses a fixed number of superpixels [7] leads to a strong trade-off between accuracy and the number of unlabeled regions. The higher the amount of superpixels, the better the performance but the greater the number of superpixels that end up unlabeled. The multi-level strategy we propose here solves these problems and improves our earlier results. Our improved approach is more robust, regardless of the modality of the input images and the sparsity of the labeling, than our previous results. We present significantly better performance and a more exhaustive validation, including baselines with more superpixel segmentation algorithms, results with new datasets having dense labels as well as an ablation study of several of the method's parameters.

2.3 Training Dense Semantic Segmentation with Sparse Pixel Labels

This section describes our approach for learning a semantic segmentation model when the available training data only has sparsely labeled pixels. Fig. 2.1 shows a summary of the main stages of our approach.

2.3.1 Problem Formulation

Performing semantic segmentation when only sparse annotations are available is a very challenging task. In this section, we formulate the problem using two different approaches. In the first approach, we crop the image into small patches, perform patch classification and then, stitch these patches back together. In the second method, we

³<https://labelbox.com/>

perform per pixel classification to directly obtain the image semantic segmentation. We will compare both approaches, focusing more on the second method.

Per patch classification. Semantic segmentation can be formulated as a patch classification problem. When a few annotated pixels are provided, a CNN can be trained on patches cropped around those labeled pixels to get a final image segmentation joining the classification result for each patch. This strategy, which has been successfully applied in existing approaches [21, 159], is trained on n labeled patches, one per annotation. The training pairs used are of the form $(\mathbf{X}_{d(i,j)}, y_{(i,j)})$ where $\mathbf{X}_{d(i,j)}$ is a patch of dimensions $d \times d$ centered around each labeled pixel with coordinates (i, j) , and $y_{(i,j)}$ is a scalar representing the label of this pixel.

Per pixel classification. More frequently, semantic segmentation is formulated as a pixel classification problem where the input and output constitute the entire image. In this case, an end-to-end CNN architecture is trained with dense labels, i.e., fully labeled images, to obtain the per-pixel classification directly, i.e., the semantic segmentation. In our case where only some sparse labels are available, there are two existing approaches for addressing the sparsity: either propagate the sparse labels into dense labels, or, train only on the sparse labels and ignore the non-labeled pixels. We previously showed [5] that the first approach provides better results as it provides more data for training.

We consider the most common fully convolutional architectures for this problem: the FCN (fully convolutional network) architecture [150], the FCN symmetric architecture [17] and the current state-of-the art, which has a light and small decoder [36]. In all these architectures, the networks are trained with pairs of images: $(\mathbf{X}, \mathbf{Y}')$, where \mathbf{X} is the original input image, an $(m \times n \times c)$ array (for an RGB image $c=3$), and \mathbf{Y}' is an $(m \times n)$ array with a label for each pixel.

Formulation. Both the per patch strategy and per pixel approach are classification problems, whose models are obtained by minimizing the error $\min(|\hat{y} - y|)$ between the predicted \hat{y} and expected values y . Both strategies are commonly optimized using the cross-entropy loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{j=1}^N \sum_{c=1}^M y_{c,j} \log(\hat{y}_{c,j}), \quad (2.1)$$

where N is the number of labeled samples (in semantic segmentation, N is the number of labeled pixels) and M is the number of classes. $Y_{c,j}$ is a binary indicator (0 or 1) of pixel j belonging to a certain class c and $\hat{y}_{c,j}$ is the CNN predicted probability of pixel j belonging to a certain class c . This probability is calculated by applying the soft-max function to the networks' output. In the per pixel approach, each j represents a pixel, while in the per patch approach, each j represents a patch, so $N = 1$ since we only have one label per patch.

2.3.2 Label Augmentation with Multi-Level Superpixels

In this section, we detail our proposed strategy for sparse labeling augmentation. The goal is not only the propagation itself but also augmenting our available sparse training data in order to boost the training and performance of CNN-based methods for semantic segmentation. Our approach for label augmentation is based on existing superpixel segmentation techniques.

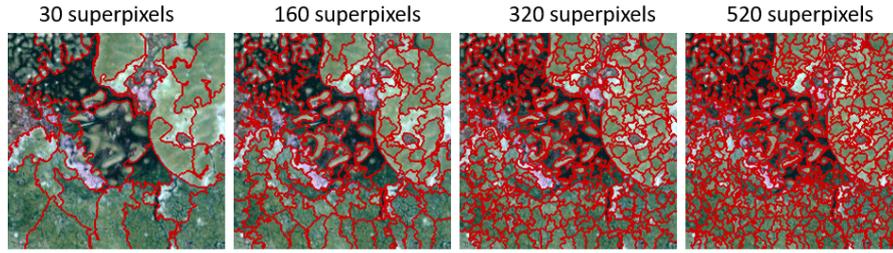


Figure 2.2: **Superpixels.** Superpixel segmentation obtained when varying the target number of superpixels (clusters). These images have been segmented using the SEEDS algorithm.

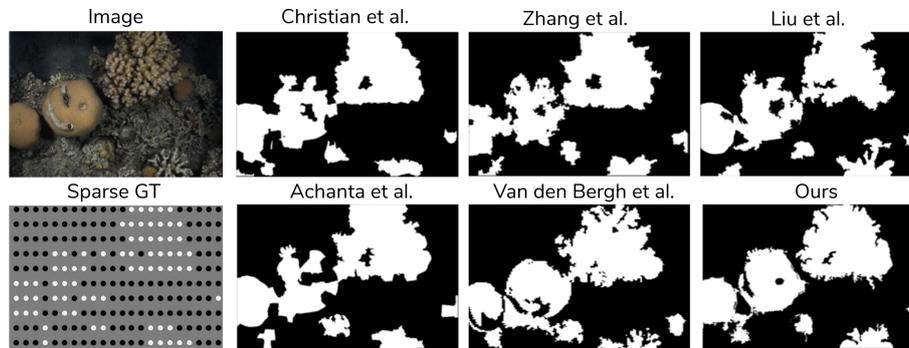


Figure 2.3: **Propagated Annotations with Superpixels.** Sparse ground truth label augmentation obtained with different superpixel segmentation techniques (black and white dots represent one single labeled pixel). The top-left view is the original image and the bottom left view is the sparse available ground truth. The rest of the images are binary (coral/no-coral) labeling augmentations.

Superpixel (single-level) based labeling propagation. Initially we consider a simple but intuitive approach: single-level superpixel-based augmentation. This strategy, detailed in our preliminary work [5], takes an input image with sparse labels and augments them in two steps. First, the image is segmented into a preset number of superpixels, as shown in the examples in Fig. 2.2. Second, the sparsely labeled pixel values are propagated following the superpixel segmentation, i.e., all pixels in each superpixel get the label value that appears the most within that superpixel. Fig. 2.3 shows some binary examples using several superpixel segmentation algorithms we evaluate in this work: Contour Relaxed Superpixels (CRS) [48], Pseudo-Boolean (PB) [277], Entropy Rate Superpixel (ERS) [149], Simple Linear Iterative Clustering (SLIC) [1] and Superpixels Extracted via Energy-Driven Sampling (SEEDS) [239]. Section 2.5.1 compares the performance of these methods in the proposed label augmentation strategy.

This single-level superpixel strategy has been used in prior works with promising results [5, 122] but has some drawbacks because the number of superpixels has to be specified *a priori*. Consequently, two issues can potentially arise. Either some superpixels may not contain any labeled pixels (and, therefore, generate unlabeled regions) or the superpixels may be too large to fit to complex or very small image shapes accurately. This leads to a strong trade-off between proper contour fit and the number

of unlabeled regions: a higher number of superpixels fits the actual shapes better, but it increases the number of superpixels that are left without any label. Our proposed multi-level strategy extension solves these problems.

Multi-level superpixel segmentation. The proposed multi-level superpixel segmentation (see Algorithm 1) consists of applying the superpixel image segmentation iteratively, progressively decreasing the number of superpixels generated in each iteration. The input of Algorithm 1 is an image, the sparse ground truth, which is an image with some labeled pixels (non-labeled pixels will have a special value) and the number of levels, which is a positive integer number and defines the number of iterations to be

Algorithm 1: Propagation with Multi-Level Superpixel Segmentation

```

1 function MLsuperpixels (SparseGT, img, n_levels)
  Input : img, i.e., the input image
           SparseGT, i.e., the corresponding sparse ground truth labeling
           nLevels, i.e., the specified number of iterations
  Output: augmentedLabeling, i.e., the augmented labeling
2 nSuperpixels  $\leftarrow$  getHighNumber()
3 augmentedLabeling  $\leftarrow$  emptyImage()
4  $i \leftarrow 1$ 
5 while  $i \leq nLevels$  do
6   superpixels  $\leftarrow$  getSuperpixels (img,
   nSuperpixels)
7   augmentation_i  $\leftarrow$ 
   getAugmentedLabels (SparseGT, superpixels)
8   augmentedLabeling  $\leftarrow$  join (augmentedLabeling,
   augmentation_i)
9   nSuperpixels  $\leftarrow$  decrease (nSuperpixels, i)
10   $i \leftarrow i + 1$ 
11 end
12 return augmentedLabeling;

```

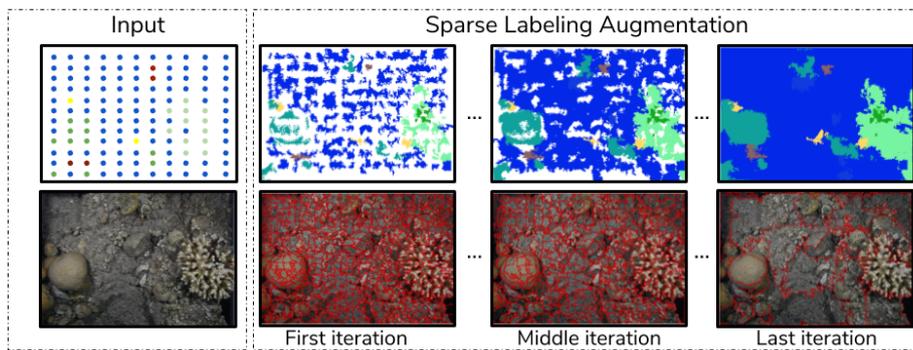


Figure 2.4: **Multi-level superpixel label augmentation algorithm.** [Left] The input of the algorithm (available sparse labels and corresponding image). [Right] The augmentation process: augmented labels (top row) after the first, middle and last iteration, and the superpixel segmentation obtained at that level (bottom row). The output of the method is the augmented labeling from the last iteration (right column).

performed.

In the first iteration, the number of superpixels is very high, leading to very small-sized superpixels for capturing small details of the images (the propagation is performed exactly as the single-level approach). The number of superpixels vis-à-vis the number of labeled pixels is automatically computed. This value can also be given as an extra parameter. In Section 2.5.1 we evaluate how this parameter affects the quality of the augmentation.

In the first iteration, as the superpixels are small, the label augmentation results in many unlabeled regions. The following iterations decrease the number of superpixels, leading to larger superpixels covering unlabeled pixels (see Fig. 2.4). Successive iterations do not overwrite information; they only add new labeling information until all pixels are covered. Parameter values for Algorithm 1 are specified in Section 2.5.1. Our code is available online⁴.

2.3.3 Semantic Segmentation Architectures and Optimization

Architectures considered. Deep learning architectures for semantic segmentation have advanced since [150] blazed a path to build different types of decoders to up-sample the learned features of the encoder. Their work uses bilinear interpolation for upsampling the last encoder layer into the output resolution. A second type of fully convolutional networks reverses the encoder architecture by constructing a symmetric architecture where the decoder has the same or similar computation as the encoder. This kind of architecture usually performs better but at a higher computational and temporal cost. SegNet [17] and FC-Densenet [112] are two examples of well-known architectures using this type of decoder.

The current state-of-the-art of semantic segmentation, Deeplabv3+ [36], follows a third and different strategy. It is based on focusing the computation on the encoder and having a light decoder that learns to decode the learned representation and requires little computation. The main features of Deeplabv3+ are the use of depth-wise separable convolutions [116], which allow convolutions to be performed with less computation and perform better when channels are decorrelated; spatial pyramid pooling [90], which allows joining of information from different resolutions in one stage; and use of dilated convolutions [270], which allows learning of complex relations between spatially separate information without the need to reduce the resolution. For our main study case, coral imagery semantic segmentation, previous work [122] has also shown that the Deeplab architectures perform better than other architectures.

In our experiments, we compare the Deeplabv3 encoder architecture with the three different types of decoders described above, to see how they affect the architecture. Thus, we compare Deeplabv3, Deeplabv3+, and Deeplabv3-symmetric. We use the official implementation for the first two architectures⁵. For the last architecture, we modify Deeplabv3+, turning it into a symmetric FCN architecture. Section 2.5.2.2 discusses the results obtained with our trained models using these three alternative architectures, both single-level and multi-level trained from scratch, as well as explores some fine-tuning options.

Loss function comparison. Apart from selecting a suitable neural network architecture, another crucial decision is selecting the loss function, as it directs the learn-

⁴<https://github.com/Shathe/ML-Superpixels>

⁵<https://github.com/tensorflow/models/tree/master/research/deeplab>

ing of the neural network. Deep learning architectures for semantic segmentation are commonly optimized using the cross-entropy loss function. Nevertheless, there are other variations, which we describe below. In this work, we propose a modification of the cross-entropy loss function that takes into account the neighboring pixels without adding much computation.

Cross-entropy loss function [53]. This is the common loss function for classification and semantic segmentation (see Eq. (2.1) in Section 2.3.1). This loss optimizes the accuracy per pixel. For classification, this fits perfectly, but for semantic segmentation, it is applied to every pixel independently and does not include information about neighboring pixels.

Lovasz loss function. Recently, a novel approach for optimizing neural networks for semantic segmentation was developed [23]. Instead of optimizing the accuracy of every pixel individually, this work tries to optimize the MIoU (Mean Intersection over Union [73]), the standard metric for semantic segmentation. One main drawback of this approach is the computation time. Computation of this loss takes around five times more than calculating the cross-entropy loss function.

Cross-entropy loss function with median frequency balancing [17]. This is a modification of the cross-entropy loss function. It consists of adding weights to every semantic class to optimize the mean accuracy per class, reducing the effect of the class imbalance. Every class c is weighted according to the following formula: $w(c) = m.f/f(c)$, where w is the weight of a class c , m is the median frequency and f is the frequency of a class c .

Our loss function. We developed a modification of the cross-entropy loss function to take into account the prediction of neighboring pixels without adding much computation. In most semantic segmentation use cases, if one pixel belongs to a certain class, its neighbors (at different distances) are likely to belong to the same class. Thus, following this intuition, we give more importance (higher loss) to pixels whose neighboring pixel predictions are not the same (we consider the pixel connectivity as 4-neighbor, i.e., 4-connectivity). By applying this idea, we achieve two main benefits:

- The loss will prevent the algorithm from predicting isolated pixels, i.e., pixels of the same type are usually together. This will help the overall accuracy and MIoU performance.
- The classes with less data will have fewer neighbors of their type; therefore, these classes will have a higher impact on the loss, correcting the class imbalance.

Following the idea of the median frequency balancing, we add some weights to every pixel p as follows:

$$w(\hat{y}) = \text{norm} \left[1 + \sum_{n=0}^N \text{gauss}(\sigma, n) (4 - \delta(\hat{y}_{i,j}, \hat{y}_{i,j+2^n}) + \delta(\hat{y}_{i,j}, \hat{y}_{i+2^n,j}) + \delta(\hat{y}_{i,j}, \hat{y}_{i-2^n,j}) + \delta(\hat{y}_{i,j}, \hat{y}_{i,j-2^n})) \right] \quad (2.2)$$

where δ is the Kronecker delta (the function is 1 if the variables are equal, and 0 otherwise), N is the number of neighboring levels to evaluate (a neighboring level n represent neighboring pixels at distance 2^n in pixels) and it is always set as the maximum possible with \hat{y} as the predicted class. We introduce the Gaussian function to force the neighbors closest to the pixel to have more impact on the weight. The σ value affects the importance that neighboring pixels are given. In two cases, all neighbors have the

same weight: when $\sigma = 0$, the multiplicative factor of all the neighbors is zero; and when $\sigma = \text{inf}$, the multiplicative factor of all the neighbors is the unity. The weight normalization (norm) consists of getting weights with mean equal to one with respect to all the predicted pixels, i.e., L1 normalization. In Section 2.5.2.1 we evaluate the effect of the parameter σ .

2.4 Datasets and Labels

This section describes the datasets and metrics used in this chapter experimentation.

2.4.1 Datasets

Table 2.1 summarizes the four datasets used for the coral segmentation experiments.

Datasets	Train images	Test images	Semantic classes	Label type	Total labeled pixels
CoralNet	416512	14556	191	Classification	431068
Eilat	142	70	10	Sparse	42400
EilatMixx	23	8	10	Sparse	5109
Mosaics UCSD	4193	729	35	Dense	1290M

Table 2.1: Details of the coral datasets used in this work.

- **CoralNet.** We processed all the CoralNet public data in order to get a useful and robust dataset, containing image crops around the sparse pixel labels having different sizes: 32×32 , 64×64 and 128×128 . We only kept the semantic classes that had at least two thousand samples. The resulting dataset consists of 431068 images. These images are from over 40 different geographical sources from around the world. Each image has at least one semantic label out of the 191 different coral species this dataset considers. We randomly selected 95% of the data for training the encoder and only 5% for testing it. The main use we make of this dataset is to train a *generic encoder for coral images* in order to learn better representations for this type of images. The source data is available at the CoralNet project website.
- **Mosaics UCSD [60].** The original dataset consists of 16 mosaics with resolution of over $10K \times 10K$. The dataset used in this work is the result of cropping these mosaics into 512×512 images, resulting in 4193 training images (85% randomly selected) and 729 test images (15% randomly selected). The dataset contains 34 different semantic classes plus the background class we ignore and provides dense labels (all pixels in each image are labeled). This dataset is used for many of our experiments due to the quantity of labeled images it has and because its labels are dense, allowing more accurate/reliable metrics.
- **Eilat [21].** This is a publicly available coral dataset⁶ consisting of 142 training images and 70 validation images. The resolution of the original images was

⁶<https://datadryad.org/resource/doi:10.5061/dryad.t4362>

3K×5K but, for our experiments, we downsized them $\times 4$ due to memory issues when feeding the CNNs. Although the labeling of this dataset is sparse and it only has few labeled pixels per image, it also has binary (coral vs. non-coral) dense labels for a subset of its images. Apart from the RGB image channels, it has two additional channels with fluorescence information.

- **EilatMixx.** This dataset consists of 31 images from the same geographical area as the Eilat dataset but acquired at a significantly different time (three years later: the Eilat dataset is from 2015 and the EilatMixx is from 2018). It contains images of the same coral species at the same resolution and with the same image processing (color correction) as the Eilat dataset. This dataset and the Eilat dataset show how challenging and heterogeneous images acquired at the same areas but at different times are. They are used in our experiments to prove that we can learn and adapt coral semantic segmentation to a new situation when having only a few sparsely labeled pixels. Both Eilat datasets contain coral images from the Red Sea (Israel). In contrast to the Eilat dataset and the CoralNet dataset, this dataset has been annotated such that specific points of interest within the image were chosen rather than having a random or uniform point grid in which not every significant object gets labeled. This dataset has fewer images than the other datasets, which is useful in our experiments as it helps to prove how the generic encoder supports learning a model for a new scenario when few training images are available.

Figure 2.5 shows some examples from all these datasets. The EilatMixx dataset is released to the community, including the new images, the original labels and our automatically augmented labels for the Eilat, EilatMixx and Mosaics UCSD datasets.

2.4.2 Reference Labels

As the datasets have either sparse or dense labels, we use different labels to evaluate the results of the segmentation models obtained, depending on the available labels.

The Eilat and EilatMixx datasets, which only provide sparse annotations, are evaluated with metrics computed using three different reference labels:

- *Original-GT*: The original sparse labels available with the dataset. This is the least representative and reliable of the three ground truth options since it has very few annotations per image, but it is necessary to perform direct comparisons with previous results that used it.
- *Augmented-GT*: The augmented ground truth obtained by our approach. This is an approximated labeling because it contains some noise. It does, however, provide a very representative reference labeling [5, 7].
- *Dense-GT*: We use this only for the Eilat dataset. It contains a few dense labeled images for binary (coral vs. non-coral) segmentation obtained by an expert coral biologist. It is only available for some images but is the most reliable and representative to use when comparing results of the semantic segmentation task.

The Mosaics UCSD dataset is the only one with dense labels. The results using this dataset are evaluated using these dense ground truth labels. As this is the most reliable evaluation, the majority of the experiments will be performed with this dataset.

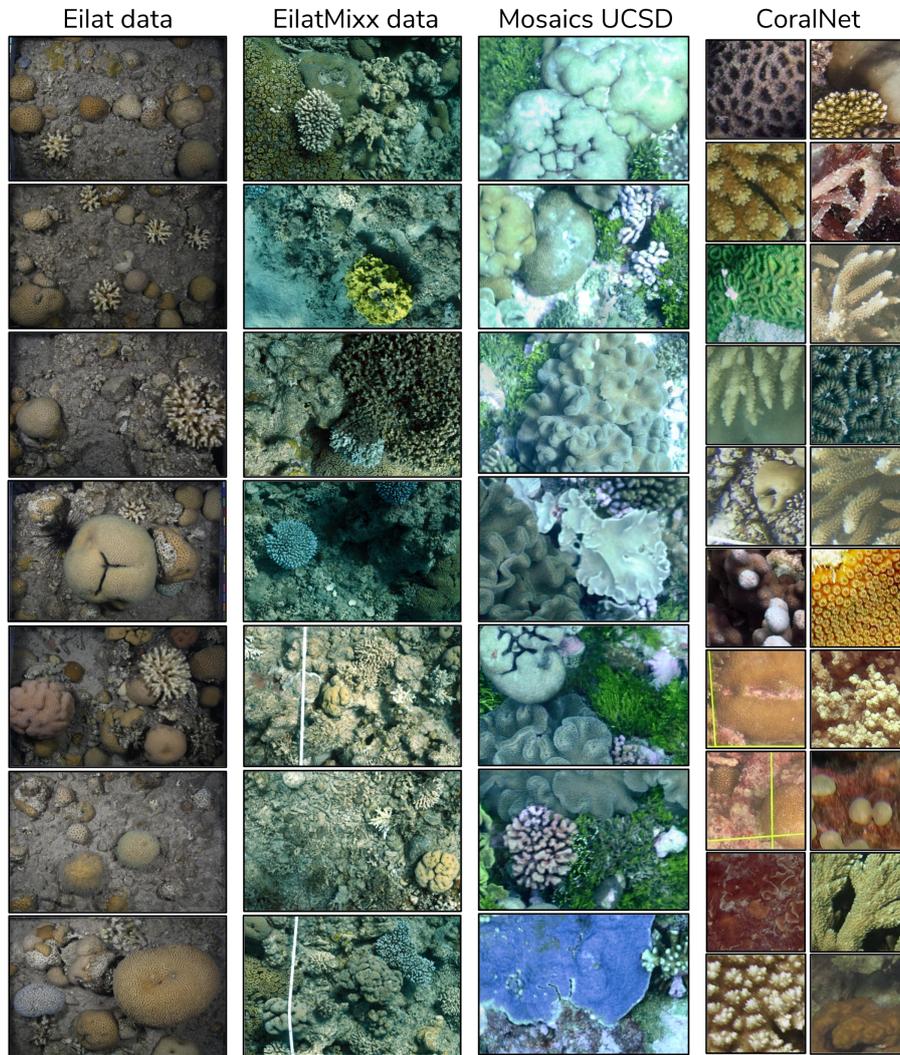


Figure 2.5: **Datasets.** Several images of the four different datasets used in this work. From left to right: Eilat [21], EilatMixx (ours), Mosaics UCSD [60] and CoralNet.

Metrics for evaluation. The metrics we use for our evaluation are the standard metrics for semantic segmentation. We just consider different types of ground truth (explained above) to compute it: **PA** – pixel accuracy; **MPA** – mean pixel accuracy (per class) and the **MIoU** – mean intersection over union.

2.5 Evaluation of the Proposed Approach

2.5.1 Labeling Augmentation with Multi-Level Superpixels

This section evaluates the labeling augmentation method detailed in Section 2.3.2.

Experiment setup. For all the following experiments, the multi-level superpixel based augmentation starts with an initial number of superpixels (init_{ns}) set to ten times the number of labeled pixels for each image. We set the final number of superpixels (final_{ns}) to the tenth of labeled pixels per image. Then, given a number of levels (NL) to complete, the number of superpixels (N_{sup}) to generate at each level (level_i) is:

$$N_{\text{sup}}(\text{level}_i) = \text{init}_{\text{ns}} \left[\left(\frac{\text{final}_{\text{ns}}}{\text{init}_{\text{ns}}} \right)^{\frac{\text{level}_i}{\text{NL}}} \right] \quad (2.3)$$

Tables 2.2 and 2.3 show the comparison between the single-level augmentation used in recent previous work [5] and other work that followed aimed at building an annotation tool [122], and the proposed multi-level augmentation (ours) using different superpixel segmentation techniques.

As Table 2.2 shows, we perform a more exhaustive comparison with the Mosaic UCSD dataset because it has dense labeling and more semantic classes. We compare the two approaches using five different superpixel segmentation algorithms (SEEDS [239], CRS [48], ERS [149], SLIC [1] and PB [277]). Our multi-level approach outperforms the single-level method by 3.76% MIoU using SLIC and by 14.11% using CRS. This is a significant improvement because the augmented labeling has to be the most accurate as possible if we want to learn a semantic segmentation model from it. Fig. 2.6 shows some visual examples, comparing the single-level and multi-level augmentations. Clearly, the multi-level algorithm outperforms the single-level method and fits the coral reef shapes better.

Regarding the Eilat dataset, the SLIC and SEEDS superpixel algorithms also outperform the ERS, CRS and PB methods. What is especially interesting about this

Augmentation Approach	Metrics		
	PA	MPA	MIoU
SEEDS single-level [5]	82.60	81.75	62.05
SEEDS multi-level (<i>ours</i>)	88.66	86.28	75.74
SLIC single-level [5]	86.93	85.72	73.20
SLIC multi-level (<i>ours</i>)	88.94	87.00	76.96
CRS single-level [5]	80.02	78.82	58.77
CRS multi-level (<i>ours</i>)	87.03	84.91	72.88
ERS single-level [5]	79.52	80.09	59.42
ERS multi-level (<i>ours</i>)	86.65	84.56	73.13
PB single-level [5]	78.66	81.02	57.41
PB multi-level (<i>ours</i>)	85.74	83.01	70.70

Table 2.2: Labeling augmentation quality when using the single-level and multi-level (15 levels) approaches. Dataset: Mosaics UCSD. Evaluation on the dense labels.

Augmentation Approach	Metrics		
	PA	MPA	MIoU
<i>Using RGB</i>			
Evaluation based on Dense-GT			
SEEDS single-level [5]	92.21	80.20	72.90
SEEDS multi-level (<i>ours</i>)	93.23	84.91	75.37
SLIC single-level [5]	92.03	81.93	73.87
SLIC multi-level (<i>ours</i>)	92.76	83.60	75.37
<i>Using fluorescence</i>			
Evaluation based on Dense-GT			
SEEDS single-level [5]	93.38	86.86	77.86
SEEDS multi-level (<i>ours</i>)	94.20	87.50	79.88
SLIC single-level [5]	93.22	84.96	77.44
SLIC multi-level (<i>ours</i>)	93.86	85.37	78.37

Table 2.3: Labeling augmentation quality when using the single-level and multi-level (15 levels) approaches on different input modalities (RGB and fluorescence images). Dataset: Eilat.

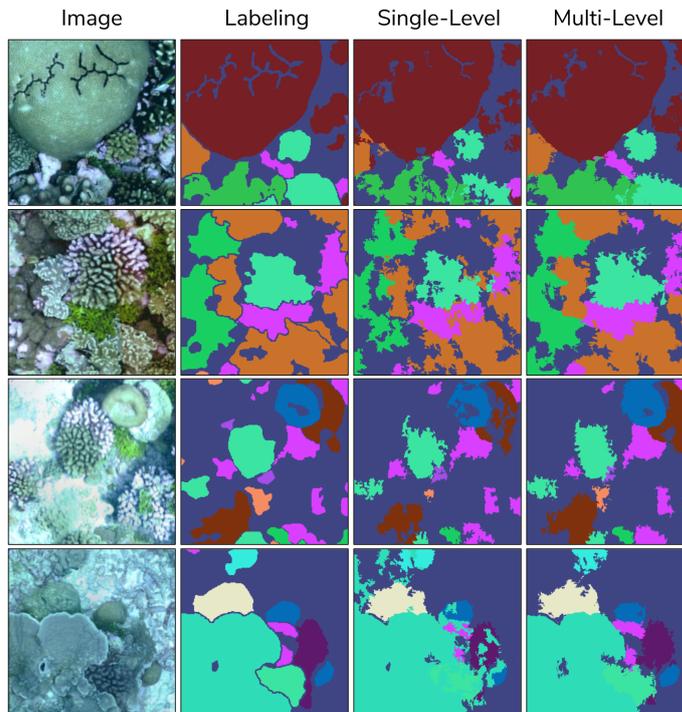


Figure 2.6: **Comparison between the single-level and the multi-level approaches.** Both are augmented from 300 labeled pixels and use the SEEDS superpixel algorithm.

dataset is the multimodal information (fluorescence) it provides. Fluorescence is a very relevant and informative source of information regarding coral reefs [21, 294]. In Table 2.3 we show how this fluorescence information can enhance the labeling augmentation process.

We perform two small experiments to show the temporal cost of our proposed

method and how the performance of our multi-level approach changes when varying the number of levels and the image resolution. Table 2.4 shows how the resolution (r) and the number of levels (n) of our multi-level algorithm affect the quality of the augmented labeling and the execution time. This experiment uses SLIC superpixels because they perform better on this dataset (see Table 2.2). Although the resolution barely affects the performance, as might be expected, it does affect the number of superpixels.

The number of superpixels considered in this evaluation increases from 1 (single-level) to 5, 15 and 30. As a result of this evaluation, we can see that as the number of superpixel increases, the accuracy of the method improves. The upper limit of the number of superpixels, at which point the accuracy starts to converge, is around 15–30 superpixels. This is why in the majority of our experiments, we use 15 superpixels as the default number for the multi-level approach. Note that our algorithm is linear in the number of levels $O(n)$ and quadratic in the resolution $O(r^2)$, i.e., linear in the number of pixels. One important improvement in our approach in this work, compared to our previous work, is the speed-up. Whereas in our earlier version [7], processing 1024×1024 pixel image required 113 seconds, in this improved version, it takes 40 seconds.

Our proposed algorithm also requires other parameters to be set, such as the initial number of superpixels. This number has to be set empirically. A high number (i.e., in the order of 10^3) is sufficient for the proper functioning of the system (see Fig. 2.2 for a visual representation of the effects of this number). We analyze the influence of varying this parameter with a small experiment. Table 2.5 shows that an initial value in the order of 10^2 works worse than one in the order of 10^3 , which is very similar to the order of 10^4 (our algorithm’s resolution, linear in the number of pixels; see above). Therefore, some thousands of superpixels are enough to capture the small details of the images. On the other hand, the final or last number of superpixels has to be set as a low number to be able to fill out and label all the pixels of the image, e.g., five superpixels.

Resolution	N-Levels			
	1	5	15	30
256x256	73.12 / 0.3	74.80 / 1.12	76.78 / 3.23	77.11 / 6.12
512x512	73.20 / 1.34	74.85 / 5.87	76.96 / 15.72	77.21 / 30.03
1024x1024	73.31 / 8.4	74.91 / 39.76	77.10 / 113.56	77.25 / 219.56

Table 2.4: Performance (MIoU / time in seconds using an Intel Core i7-6700) when varying the number of levels in the labeling augmentation and the image resolutions. Experiment performed on Mosaics UCSD dataset. Evaluation on the dense labels. Sparsity used as input: 0.1% of the labeled pixels (300 pixels).

N superpixels	MIoU
$\times 100$ the number of labeled pixels (30000)	77.07
$\times 10$ the number of labeled pixels (3000)	76.96
$\times 1$ the number of labeled pixels (300)	74.87

Table 2.5: Performance (MIoU) when varying the number of superpixels in the first level of our algorithm. Experiment performed on Mosaics UCSD dataset. Evaluation on the dense labels. Sparsity used as input: 0.1% of the labeled pixels (300 pixels). We use 15 levels for this experiment.

2.5.2 Analysis of Semantic Segmentation Methods

This section discusses the semantic segmentation architectures and losses described in Section 2.3.3.

2.5.2.1 Efficient Semantic Segmentation

This experiment compares the performance of different common losses for semantic segmentation including our proposed modification of the cross-entropy detailed in Section 2.3.3.

Experiment setup. To perform a fair comparison, for all the executions we use the same semantic segmentation model: Deeplabv3+ [36]. We train it for 600 epochs with an initial learning rate of 10^{-3} with a polynomial learning rate decay schedule. During the training, we perform data augmentation: vertical and horizontal flips, contrast normalization, and random image shifts and rotations. For this experiment, we use the Mosaics UCSD dataset because it has dense annotations that facilitate a fair evaluation.

Loss function comparison. Table 2.6 shows a comparison between the most common losses used in semantic segmentation using deep learning and our proposed modification of the cross-entropy loss. The level of performance of the functions is close; however, our modification performs slightly better than the cross-entropy loss for the most important metrics for semantic segmentation. In contrast, the median frequency balancing performs better for mean accuracy, as might be expected, having a negative effect on the accuracy per pixel and on the MIoU. Our proposed modification has no negative effect on any of the metrics. Analyzing its properties in more detail, we see that increasing the number of neighboring pixels to take into account ($\sigma > 0$) increases the performance. We also see that giving less weight to far neighboring pixels ($\sigma < \text{inf}$) also has a positive effect on the performance. In this experiment we set $\sigma = 3$, as an example of $0 < \sigma < \text{inf}$. We empirically found that values $2 < \sigma < 5$ work very similarly. Regarding the time for performance, using the Mosaics UCSD dataset, one epoch takes almost 8 minutes, but the Lovasz loss takes 37 minutes per epoch, which is almost five times more than the other losses.

Loss configuration	Metrics		
	PA	MPA	MIoU
Cross-entropy [53]	85.31	55.78	45.60
Median freq. balancing [17]	82.11	61.96	43.02
Lovasz [23]	85.15	59.91	47.28
Ours ($\sigma = 0$)	85.54	58.17	47.59
Ours ($\sigma = 3$)	86.11	59.90	49.16
Ours ($\sigma = \text{inf}$)	85.97	59.72	48.76

Table 2.6: Semantic segmentation performance using different loss functions for training. Experiment performed on Mosaics UCSD dataset. Evaluation on the dense labels.

Architecture	Metrics					
	PA	MPA	MIoU	GPU Time	GFlops	Params
Deeplabv3 [35]	85.72	58.73	48.41	22ms	48.80	40.89M
Deeplabv3+ [36]	86.11	59.90	49.16	26ms	51.44	41.05M
Deeplabv3-symmetric	87.16	61.12	51.57	41ms	65.63	43.33M

Table 2.7: Semantic segmentation performance of different architectures. GPU time is the inference time on a Titan XP GPU. Experiment performed on Mosaics UCSD dataset. Evaluation on the dense labels.

2.5.2.2 Semantic Segmentation Architectures

This experiment compares the performance of the different common architectures for semantic segmentation detailed in Section 2.3.3.

Experimental setup. To perform a fair model comparison, we use the same configuration for all models. The training configuration is the same as in the previous experiment with the exception that we use the same loss: our modification of the cross-entropy. We use the Mosaics UCSD dataset for this experiment because it has dense annotations that facilitate a fair evaluation. The batch size is set to 8, except for the Deeplabv3-symmetric (batch size of 6) due to memory issues.

Architecture comparison. Table 2.7 shows the performance comparison of different Deeplabv3-based architectures, i.e., the same state-of-the-art encoder with different decoder options to achieve the segmentation (more details are given in Section 2.3.3). The performance gap between the Deeplabv3 and Deeplabv3+ models is small in our case, compared to the larger increases observed in prior work using other datasets [36]. The results using our modification of Deeplabv3-symmetric show that the symmetric architecture performs better, but demands a noteworthy increase in the computation and inference times. The symmetric architecture has a larger decoder that is able to learn how to decode the features better. One possible problem of such a deep architecture is the vanishing gradient problem, but the skip connections between the early layers of the encoder and the later layers of the decoder solve this problem. As the symmetric architecture has more convolutional layers and, therefore, more parameters to learn, this architecture performs slightly better than the other architectures. Nevertheless, some applications may not be able to afford the additional computation and time costs.

2.5.3 Training with Augmented Labels

This experiment aims to answer one of the main research questions of this work: *Can we get a semantic segmentation model trained from sparse labels that is similar to one trained using dense labels?*

Experimental setup. To answer this question, we compare the semantic segmentation results of a model trained on dense labels and models trained on our augmented labels from sparse labels. We trained the Deeplabv3-symmetric architecture (the one that performed best in Section 2.5.2) with the dense labeling and two different augmented labeling setups: augmented labeling from 300 labeled pixels (0.1% of the dense labels)

and with only 30 labeled pixels (0.01% of the dense labels). Regarding the augmentation process, we set the number of levels to 15. These three models are evaluated with the dense labels. To perform a fair model comparison, we use the same configuration for all models. The training configuration is the same as the previous experiment with the exception that here we use the same loss, our modification of the cross-entropy that gives the best results.

Results on the Mosaics UCSD dataset. The results shown in Table 2.8 suggest, as expected, that having more labeled pixels, the results improve. Nevertheless, training with only some labels and augmenting them with our approach leads to similar performance while significantly reducing the labeling annotation cost. The main reasons for the great performance of our method are that neural networks can learn and generalize representations even with some noise in the labels [226] and that our augmented labeling as shown in Table 2.2 and Fig. 2.6 is fairly similar to the dense labels (superpixel techniques adjust quite well to object edges).

We show that with our modified architecture (Deeplabv3-symmetric) and training with our augmented labeling from 30 pixels (Table 2.8), we get the same results as training with Deeplabv3 with the dense labels (Table 2.7). We also get even better performance when training with our Deeplabv3-symmetric architecture and the augmented labeling from 300 pixels than when training with Deeplabv3+ and the dense labels. One thing to take into account in our labeling augmentation approach is that its performance depends on how detailed the dataset is. This means that the more objects in the images, and the smaller they are, the more difficult to augment the labeling. In other words, our method needs to have at least one labeled pixel per object/instance in the image to be able to properly augment the labeling.

For the multi-level augmentation, we evaluated other potential improvements, which did not improve the augmented labeling results. The most interesting modification studied is weighting the loss corresponding to different augmentation levels differently. The intuition is that the augmented labels near the seeds (the sparse labels from which we augment) should have more impact on the loss because they should be more reliable and have a higher probability of being correctly labeled. The experiment results, however, did not show significant improvements.

Results on the Eilat dataset. Regarding the Eilat dataset, we compare our approach with prior work published by the authors of the dataset for multi-class semantic segmentation. The authors [21] perform a patch-based classification approach (explained in Section 2.3.1, the same approach that other works have followed [159]). We also

Trained on	Metrics		
	PA	MIoU	MPA
Dense labels	87.16	61.12	51.57
Augmented labels (300 labeled pixels)	86.30	60.00	49.93
Augmented labels (30 labeled pixels)	84.10	59.19	48.73

Table 2.8: Semantic segmentation performance of different training approaches: Training with dense labels, augmented labels (from 300 labeled pixels) and augmented labels (from 30 labeled pixels). The experiment used the Mosaics UCSD dataset. Evaluation on the dense labels.

Method	Metrics		
	PA	MPA	MIoU
Evaluation on dense scores: Augmented-GT			
Patch-based v1 [21]	—	—	—
Patch-based v2 [21]	73.61	25.32	17.89
Baseline [5]	85.88	42.25	31.12
Ours	90.02	47.61	40.65
Evaluation on sparse scores: Original-GT			
Patch-based v1 [21]	87.80	48.50	—
Patch-based v2 [21]	90.20	53.10	43.66
Baseline [5]	81.23	41.97	28.14
Ours	84.80	54.65	44.01

Table 2.9: Semantic segmentation performance when training from sparse labels. The experiment used the Eilat dataset.

compare our approach to our baseline and previous work [5].

Table 2.9 summarizes these results. We compare results from [21] (Patch-based v1) with our implementation of it using a newer CNN model (Patch-based v2). Note that (v2) performs the same as or better than the original (v1), and that (v1) is shown only where the original publication included results. Results also include our previous work (Baseline) with the single-level label augmentation [5], and our work presented here (Ours). We show the original-GT scores because some related work has published results using this. Note, however, how the proposed method significantly outperforms previous work on the more significant dense scores.

2.6 A generic pre-trained coral encoder

2.6.1 Pre-Training and Fine-Tuning

In this section, we study how to train models for coral segmentation that can generalize to other regions or across time.

Pre-training deep learning models on large general datasets and then fine-tuning for more specific tasks is a widespread practice that improves deep learning performance [136], especially when large amounts of labeled data are not available. It consists of training the model on a large database of a similar domain and using that trained model as an initialization for training with the specific task data. This pre-training generalizes the final model and prevents overfitting when the specific training data is not large enough or heterogeneous. The fine-tuning of a pre-trained model can be carried out in different ways, including adjusting the number of layers vis-à-vis the original model. This process depends mostly on how different the pre-trained domain is from the target domain (the more different, the more layers we need to adjust) and how much labeled data from the target domain are available (the fewer data we have, the fewer number of layers we would typically fine-tune).

We built a generic model using a large set of coral reef data from many different locations. Our pre-trained encoder is the equivalent of what is commonly done with general purpose detection and classification, through pre-trained encoders on Imagenet [56], but ours is specifically for corals. One of the largest existing sources of

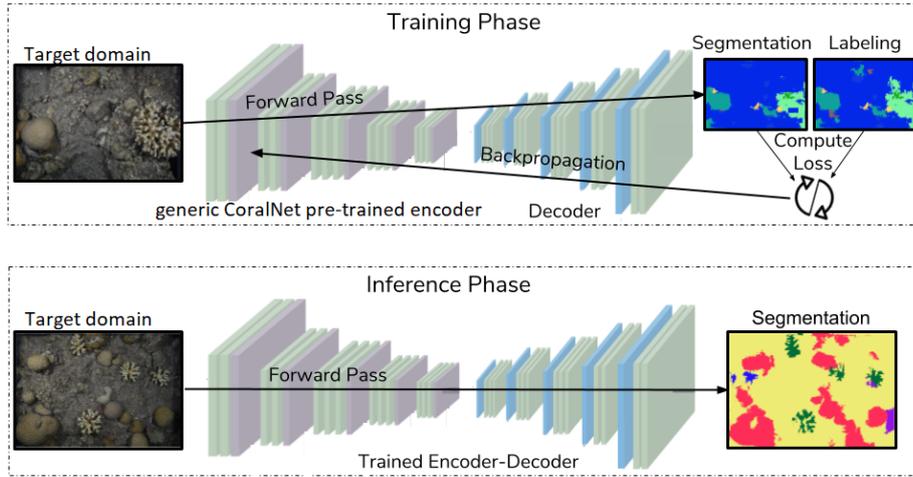


Figure 2.7: **Training and inference phases.** In the training phase, the encoder used is the generic encoder pre-trained on CoralNet. The training is performed with the target domain data using the augmented labeling. The inference phase provides the semantic segmentation result from the target domain with data for a new scenario using a generic pre-trained encoder.

coral data, CoralNet [20], is a resource for benthic images analysis and also serves as a repository and collaboration platform. In cooperation with the CoralNet team, we extracted and cleaned their public data to get a useful and robust dataset. This dataset consists of 431068 images of 191 different coral species (see Table 2.1). Its training set has between one and 2500 images per coral reef class, and the test set has up to one hundred images per coral reef class.

We trained the encoder used on the three Deeplabv3 architectures using this CoralNet dataset (see Fig. 2.7).

Note that training a general semantic segmentation was not feasible due to lack of data and difficulty in generalizing coral appearances. Since having a general segmentation model that contains all possible classes of interest on all the coral reef scenarios is the objective, our goal is to provide a generic encoder that has learned good features representing this kind of underwater imagery. Coral segmentation models for specific new scenarios can benefit from this pre-trained encoder. In the following experiments we demonstrate two main benefits of using this pre-trained generic encoder we have now made available: better performance and faster convergence.

Our semantic segmentation approach learns mostly the different colours and textures between the different coral species, since the model used captures chiefly this kind of visual local features rather than shape [76]. Nevertheless, as we trained our encoder on two hundred different coral species, where same species with different morphology are actually annotated as different semantic classes, the resulting segmentation model is also learning implicitly some of the morphological differences.

Loss configuration	Metrics			
	PA	MIoU	MPA	Epochs to converge
Mosaics from scratch	87.16	51.57	61.12	500
Mosaics pre-trained on CoralNet	87.82	53.63	63.74	300
Eilat from scratch	90.02	40.65	47.61	600
Eilat pre-trained on CoralNet	90.17	42.45	50.65	300

Table 2.10: Semantic segmentation performance of models trained from scratch and pre-trained on the CoralNet dataset. The experiment was performed on the Mosaics UCSD dataset (evaluated with dense labels) and Eilat dataset (evaluated with Augmented-GT).

2.6.2 Experiments

The aim of this experiment is to learn a good feature encoder that is able to generalize on the basis of several coral reefs species in order to be used as a pre-trained model for training on other coral datasets.

Set up. We trained the Deeplabv3 encoder from scratch on the CoralNet dataset for 70 epochs. We set an initial learning rate of 10^{-3} with a polynomial learning rate decay schedule. Our data augmentation included: vertical and horizontal flips, contrast normalization, and random shifts and rotations. For the semantic segmentation experiments, we used the better setup so far, with our proposed modified loss and the Deeplabv3-symmetric architecture.

Trained encoder. The resulting trained encoder learned a balanced feature encoding of the coral domain. The mean accuracy per patch (over the 431068 patches) of the model is 53.64, the mean accuracy per class (over the 191 different semantic classes) is 50.87 and the mean precision per class is 52.53. This result shows that the encoder has learned useful representations for the coral reef images with no class imbalance.

Benefits of the pre-trained CoralNet encoder. Table 2.10 shows the effect of the pre-trained encoder on the Mosaics UCSD dataset, which is a medium-sized dataset of four thousand training images and on the Eilat dataset, which is a small dataset of one hundred training images. The pre-training shows two main benefits. The earlier convergence on both datasets and the improved performance of both datasets. This experiment shows the power of pre-training on deep learning. The CoralNet pre-trained encoder we release will be useful for all the coral reef semantic segmentation models. Moreover, as all the deep learning classification architectures are encoders, this pre-trained encoder would also benefit coral reef classification tasks. This experiment shows the results without freezing any layer and training all the network. Other experiments performed showed that freezing layers did not help the performance.

From Eilat to EilatMixx: Generalizing to the same coral domain. Having demonstrated that through pre-training and fine-tuning we can learn more general and better models, another question may arise: *Can a learned model be used for the same domain but in different images or datasets without the need for re-training?* This question is

Loss configuration	Metrics		
	PA	MIoU	MPA
Evaluation on dense scores: Augmented-GT			
Random initialization (no training)	8.32	2.11	9.56
Eilat trained model (no training)	23.54	5.10	11.46
From scratch	46.73	10.13	16.55
Pre-trained on Eilat	44.36	10.39	16.67
Pre-trained on CoralNet	44.07	12.45	21.27
Evaluation on sparse scores: Original-GT			
Random initialization (no training)	8.39	5.78	11.13
Eilat trained model (no training)	29.02	8.21	15.24
From scratch	46.45	10.62	17.52
Pre-trained on Eilat	48.19	12.68	19.74
Pre-trained on CoralNet	49.71	14.61	25.86

Table 2.11: Semantic segmentation performance of different training approaches, including no training on the target EilatMixx data. Experiment performed on EilatMix dataset.

very interesting because it opens up the possibility for learning general models capable of being trained only once and then used for different applications for the same domain.

A quick experiment is enough to show that the answer is that it is very difficult because of coral reef variability over time and over different geographical areas (shape, sizes, color, appearance) [288] and that model fine-tuning is essential for achieving good segmentation results.

As detailed in Section 2.4, the EilatMixx dataset contains the same types of corals as the Eilat dataset and both datasets are from the same geographical area. In this short experiment, we compare how a model trained on the Eilat dataset performs on the EilatMixx dataset without any training. We compare this method with different training approaches on the EilatMixx dataset: from scratch, pre-training on the Eilat dataset and using the pre-trained CoralNet encoder.

Table 2.11 shows that the worst segmentation results are obtained with no training on the new data of EilatMixx. Although the model trained on Eilat does not reach satisfactory segmentation results on EilatMixx data, it is better than just a random solution (obtained by the mean of 10 executions with random initialization of the CNN)—which means that the Eilat data has helped to learn useful features for the EilatMixx data.

Better results are obtained after training a model on the target dataset, the EilatMixx. Moreover, the models pre-trained on other coral reef datasets achieve better results. This is the same conclusion as that obtained with the previous experiment on the Mosaics UCSD dataset and the Eilat dataset (see Table 2.10). One interesting point to consider regarding pre-training is the following: the amount of pre-training data is more relevant than having data from a very close domain for pre-training, i.e., pre-training on CoralNet, very large but not that similar to EilatMixx as Eilat, is the best performing option.

Figure 2.8 shows some visual results for the three coral reef datasets we use to get the semantic segmentation. We can see that the augmented labeling fits the coral reef images reasonably well and that the semantic segmentation obtained is good even though it has been learned from sparse labels and from a very low number of images.

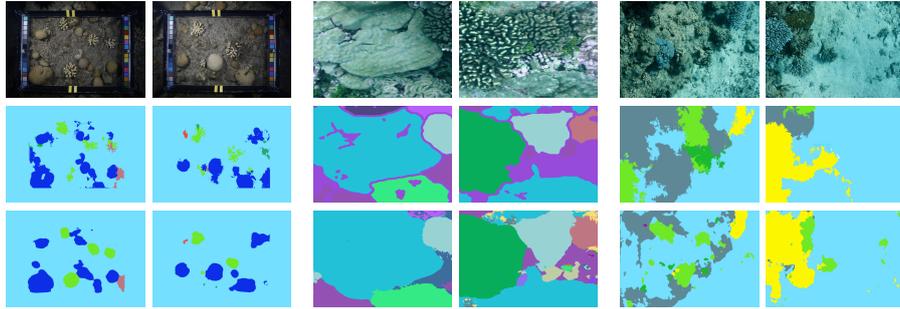


Figure 2.8: **Visual results.** Visual samples of the Eilat dataset (left), Mosaics UCSD dataset (center) and the EilatMixx dataset (right). The first row (top) corresponds to the RGB image, the center row corresponds to the augmented labeling obtained with our approach and the last row (bottom) corresponds to the semantic segmentation obtained with a model trained on the augmented labeling.

2.7 Applicability to Non-Coral Domains

We demonstrated in previous sections that our proposed method for sparse labeling augmentation allows the training of coral reef semantic segmentation models as if training with dense labels. Other domains also suffering from lack of dense labels for semantic segmentation may benefit from our method or can take advantage of the reduction in annotation cost offered by it. This section demonstrates that our proposed method can be applied to other domains.

2.7.1 Data and Evaluation

Datasets. For evaluating our labeling augmentation method, we use three datasets from different domains and with assorted objectives: the Camvid dataset (urban scenarios), RIT dataset (drone views) and VOC-2012 dataset (general purpose images).

- **Camvid** [29] is an autonomous driving dataset with 11 different classes, frequently used to train existing state-of-the-art approaches for urban area image segmentation models.
- **RIT** [119] is an aerial imagery dataset with multi-spectral data from 18 classes. RIT does not provide test image labeling, so we evaluate its results by separating part of the evaluation set it provides.
- **Pascal VOC 2012** [61] is a well-known general-purpose dataset for semantic segmentation with 20 different classes.

Evaluation. All these datasets have dense labels and, therefore, the evaluation metrics are computed with respect to these dense labels. The sparse labels of these datasets are obtained automatically by sampling the dense labeling following a grid. The default of this simulated sparse labeling is 0.1% of the dense labels (e.g., from a 500×500 image, the simulated sparse ground truth contains 250 labeled pixels). We use the same metrics as in the previous evaluations (PA, MPA, and MIOU).

2.7.2 Approach Performance on Additional Domains

Labeling augmentation quality. In this experiment, we compare the dense labels available on each dataset and the results from applying our approach to augment the *simulated* sparse labeling.

Table 2.12 summarizes the quantitative comparison of the augmented labeling with the original dense labeling (augmentation from the 0.1% of the dense labels), showing very good results in the three different domains. As noted in Section 2.3.2, our proposed augmentation method propagates existing sparse labels; therefore, it needs to have at least one labeled pixel per object or instance. The sparse labeling simulation (sampling) can miss samples from very small instances. Consequently, the PASCAL VOC 2012, the dataset with bigger and fewer objects (see Fig. 2.9), gets the highest scores. We show that the augmented labeling obtained with our approach is very close to the original dense labels. Fig. 2.9 shows qualitative results of these experiments. We can see that although our approach is not perfect and introduces some noise on the labels, it gets satisfactorily similar dense labels.

Table 2.13 compares our approach using different sparsity levels (different numbers of labeled pixels for the augmentation), with other recent label augmentation or propagation methods using the PASCAL VOC 2012 dataset. One of these works [245] uses traces as the input of the augmentation process (Traces) as well as the learned boundaries (learned by a neural network) using the RAWKS algorithm (v1) to augment the trace sparse labeling. V2 indicates the evaluation is done on 94% of the pixels, where the model is confident enough. Our baseline and previous work [5] use the single-level version of our approach and the same grid structure of sparse pixels as our multi-level superpixel augmentation. We show that our approach gets the highest scores when the input labeled pixels are more than the 0.1% of the dense labels.

Training with augmented labels. In this experiment, we compare the quality of the segmentation obtained from a model trained on the original dense labeling and from a model trained on the augmented labeling using our augmentation method.

Table 2.14 shows a summary of the results using the Camvid and RIT datasets,

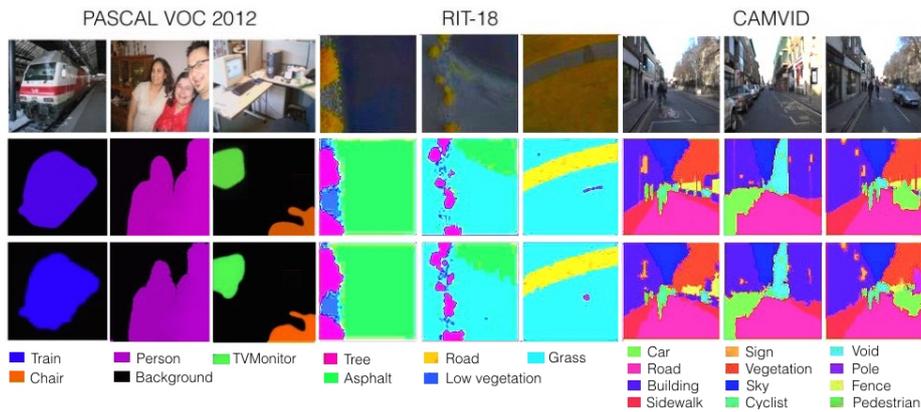


Figure 2.9: **Labeling augmentation results.** Examples of labeling augmentation evaluation with different datasets. Input images (top), original dense labeling (middle), and augmented labeling recovered from just 0.1% of the original labeled pixels (bottom).

Datasets	Metrics		
	PA	MPA	MIoU
Camvid	91.95	76.91	65.05
RIT	97.44	72.31	59.18
VOC 2012	96.87	95.77	93.31

Table 2.12: Labeling augmentation quality of our proposed method. Evaluation on the original dense labels.

Augmentation from traces	MIoU
Traces (SPCON) [245]	76.50
Traces (RAWKS v1) [245]	75.80
Traces (RAWKS v2) [245]	81.20
Augmentation from sparse pixel labels	MIoU
Baseline from 0.1% of pixels (300 pixels) [5]	86.36
Ours from 0.01% of pixels (30 pixels)	74.40
Ours from 0.1% of pixels (300 pixels)	93.31
Ours from 1% of pixels (3000 pixels)	97.25

Table 2.13: Labeling augmentation quality of different approaches. Experiment performed on the PASCAL VOC 2012 dataset. Evaluation on the original dense labels.

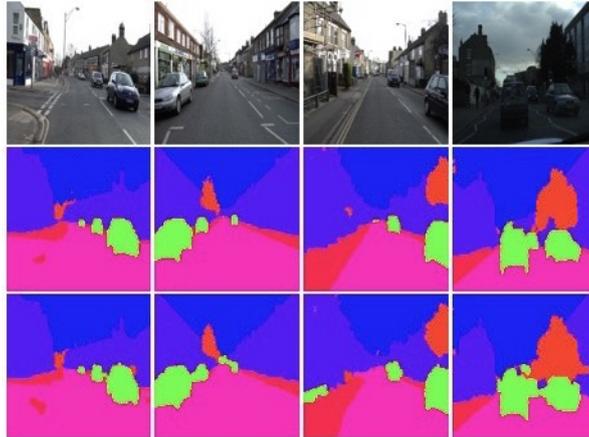


Figure 2.10: **Semantic segmentation results.** Semantic segmentation on Camvid. Original images (top), results using a model trained on original dense labeling (middle), and results using a model trained with our proposed augmented labeling (bottom).

which are the two datasets that obtained the lower augmentation scores in Table 2.12. The results obtained after training with our augmented labels are comparable to training with the original dense labels. This could be expected since we already validated that the augmented labeling is very close to the original labeling. Fig. 2.10 shows the visual comparison between the semantic segmentation results obtained with the model trained with dense labels and the model trained with augmented labels.

The conclusions are the same as the ones obtained with the coral reef data, proving our method to be both applicable to and valuable for other domains and applications.

Datasets	Metrics		
	PA	MPA	MIoU
Camvid (dense)	88.68	48.81	44.36
Camvid (augmented)	87.70	46.97	42.95
RIT (dense)	94.23	20.36	19.16
RIT (augmented)	89.30	19.65	17.85

Table 2.14: Semantic segmentation performance when training on the original dense labels (dense) and our augmented labeling (augmented). Experiment performed on the Camvid and RIT datasets. Evaluation on the original dense labels.

2.8 Conclusion

Existing acquisition systems, such as autonomous robots or remote-controlled platforms, have made it possible to acquire large amounts of environmental monitoring data, but methods to automatically process these huge amounts of data remain an open challenge in many domains. The contributions presented in this work help tackle this challenge, especially when there are not enough resources to label large amounts of detailed training data. The new tools provided by our work enable further work on scene understanding for numerous robotics applications such as remote monitoring from UAVs or underwater devices.

Our main contribution is an approach to enable effective training of semantic segmentation models from sparsely labeled data. Our approach propagates the sparse labels (sparse pixel annotations) by an iterative method based on superpixel segmentation techniques. Our multi-level approach outperforms by an average of 11% of the MIoU compared to previous single-level approaches, including our earlier version of this work. The exhaustive experimentation presented here shows the effect of the various method parameters.

The limitations of our approach come from the trade-off between number of levels in the segmentation versus computational cost. A higher number of levels yields better performance but considerably increases execution time. Another limitation to consider is that our approach relies on superpixel techniques; therefore, the image has to have clear gradients for good performance. The results in this work demonstrate that our propagated labels are highly reliable for training, as the semantic segmentation models trained with them result in performance equivalent to training with ground truth dense labels (fully annotated images).

Our core experimentation was run on a realistic and challenging scenario of underwater coral reef monitoring data. Besides the well-known environmental value of these underwater regions and consequent interest in their monitoring, they present a challenging and real-world use case where most of the available labeling efforts, made by marine biology experts, consist of sparse labels. Although the experimentation in this work is focused on underwater imagery, we also demonstrated the applicability of our approach to different applications with data from different robotic acquisition platforms (aerial surveillance and urban driving scenarios).

Further, this work contributes to the field of automatic underwater image processing as follows. We present a comparison of the main semantic segmentation architectures run in an underwater domain, in particular, coral reef image segmentation. We not only present a detailed comparison of common architectures and loss functions for the coral reef segmentation use case, but also propose a more suitable variation of the

cross-entropy loss for this task. We observed that our modified version of the cross-entropy loss enhances the results by 2% of the MIoU. Our experiments demonstrate that this encoder helps segmentation models for new coral reef scenarios, having little training labeled data available, learn better. Specifically, we are able to train models in half the time (early convergence) and enhance results by 4% MIoU. We also show that when using our pre-trained encoder, we get better results than pre-training the encoder with data from the same geographical localization. This work releases several useful tools for the research community, namely, the obtained generic encoder pre-trained on over half a million images of corals, all the data including new labeled data for coral segmentation, and the tools developed (to facilitate replication and training on new coral data).

We aim to expand our study and proposed pipeline to 3D input data, since many of the monitoring acquisition systems now provide 3D scans of the environment, rather than regular images.

Chapter 3

Semi-Supervised Semantic Segmentation

As discussed in previous chapter, annotations for semantic segmentation are expensive. Therefore, being able to learn with fewer labels can potentially provide lots of benefits such as better performance with the same annotation cost or, similar performance with fewer annotation efforts. This chapter follows the same challenge from Chapter 2, lack of labeled data, but tackling the semi-supervised scenario. In semi-supervised learning, instead of having sparse labels like in the previous chapter, only a small amount of the available data is fully labeled, and the rest of the data remains unlabeled. Therefore, the goal of this chapter is to learn a semantic segmentation model by extracting the knowledge from the few labeled samples and, from the unlabeled samples.

3.1 Introduction

The goal of semantic segmentation consists in assigning a semantic class label to each pixel in an image. It is an essential computer vision task for semantic scene understanding that plays a relevant role in many applications such as medical imaging [205] or autonomous driving [16].

As for many other computer vision tasks, deep convolutional neural networks have shown significant improvements in semantic segmentation [11, 16, 112]. All these examples follow supervised learning approaches, which require a large set of annotated data to be able to generalize well. However, the availability of labeled data is a common bottleneck in supervised learning, especially for tasks such as semantic segmentation, which require tedious and expensive per-pixel annotations.

Semi-supervised learning assumes that only a small subset of the available data is labeled. It tackles this limited labeled data issue by extracting knowledge from unlabeled samples. Semi-supervised learning has been applied to a wide range of applications [240], including semantic segmentation [67, 107, 172]. Previous semi-supervised segmentation works are mostly based on per-sample entropy minimization [107, 137, 183] and per-sample consistency regularization [67, 183, 231]. These segmentation methods do not enforce any type of structure on the learned features to increase inter-class separability across the whole dataset. Our hypothesis is that overcoming this limitation can lead to better feature learning. In particular, we expect to

learn better from the unlabeled data, which is critical when the amount of available labeled data is low.

This work presents a novel approach for semi-supervised semantic segmentation. Our approach follows a teacher-student scheme whose main component is a novel representation learning module (see Figure 3.1). This module is based on contrastive learning [87] and enforces the class-separability of pixel-level features. To achieve this, the teacher network produces feature candidates, only from labeled data, to be stored in a memory bank. Meanwhile, the student network learns to produce similar class-wise features from both labeled and unlabeled data. The features introduced in the memory bank are selected based on their quality and on their learned relevance for the contrastive optimization. In addition to increased inter-class separability, the module enforces the alignment of unlabeled and labeled data (memory bank) in the feature space, which is another unexploited idea in semi-supervised semantic segmentation.

The effectiveness of the proposed approach is demonstrated on well-known benchmarks for semi-supervised semantic segmentation, reaching the state-of-the-art on different set-ups. Additionally, our approach can naturally tackle the semi-supervised domain adaptation task, also obtaining state-of-the-art results. In all cases, the improvements upon comparable methods increase with the percentage of unlabeled data. The detailed ablation study performed shows the significance of the different components of the proposed approach.

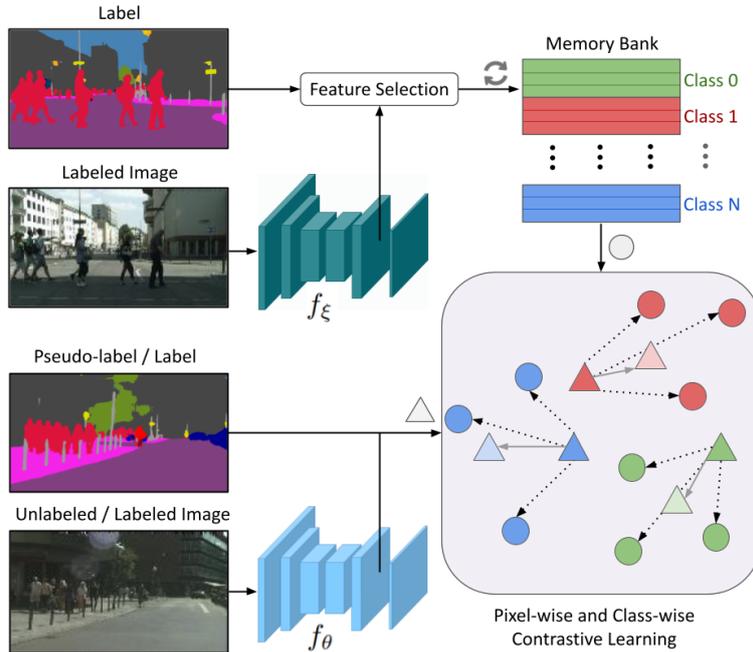


Figure 3.1: **Proposed contrastive learning module overview.** At each training iteration, the teacher network f_ξ updates the feature memory bank with a subset of selected features from labeled samples. Then, the student network f_θ extracts features Δ from both labeled and unlabeled samples, which are optimized to be similar to same-class features from the memory bank \circ .

3.2 Related Work

This section summarizes relevant related work for contrastive learning and semi-supervised learning, with particular emphasis on work related to semantic segmentation.

3.2.1 Contrastive Learning.

The core idea of contrastive learning [87] is to create positive and negative pairs from data, to attract the positive and repulse the negative pairs in the feature space. However, recent works [44, 83, 102] have shown similar level of performance using positive pairs only. The main difference in current methods is how to obtain these pairs: by using a memory bank [259], by using a momentum model [43] or directly from the same batch [42]. Contrastive learning has been recently popularized for self-supervised representation learning [43, 83, 259, 268]. As for semantic segmentation, contrastive learning has been mainly used as pre-training [250, 261, 263]. Very recently, Wang et al. [249] have shown improvements in supervised scenarios applying contrastive learning in a pixel and region level for same-class supervised samples. Van et al. [241] have shown the advantages of contrastive learning in unsupervised set-ups, applying it between features from different saliency masks.

In this work, we propose to use contrastive learning to boost the performance in semi-supervised semantic segmentation tasks. Differently from previous works, our contrastive module aligns features from both labeled and unlabeled data to high-quality features from all over the labeled set that are stored in a memory bank. We follow the positive-only research branch for computational efficiency.

3.2.2 Semi-Supervised Learning

This section discusses the two most common strategies for semi-supervised learning, pseudo-labeling and consistency regularization, as well as the application of semi-supervised learning to semantic segmentation.

Pseudo-Labeling. Pseudo-labeling leverages the idea of creating artificial labels for unlabeled data [164, 213] by keeping the most likely predicted class by an existing model [137]. The use of pseudo-labels is motivated by entropy minimization [82], encouraging the network to output highly confident probabilities on unlabeled data. Both pseudo-labeling and direct entropy minimization methods are commonly used in semi-supervised scenarios [64, 117, 183, 220] showing great performance. Our approach makes use of both pseudo-labels and direct entropy minimization.

Consistency Regularization. Consistency regularization relies on the assumption that the model should be invariant to perturbations, e.g., data augmentation, made to the same image. This regularization is commonly applied by using two different methods: distribution alignment [24, 209, 231], or augmentation anchoring [220]. While the distribution alignment enforces the predicted class distributions of perturbed images to be the same as the non-perturbed image class distribution, the augmentation anchoring forces the perturbed images to be classified as the non-perturbed image. While distribution alignment enforces the perturbed and non-perturbed to have the same class distribution, augmentation anchoring enforces them to have the same semantic label. To produce high-quality non-perturbed class distribution or prediction on unlabeled

data, the Mean Teacher method [231], proposes a teacher-student scheme where the teacher network is an exponential moving average (EMA) of model parameters, producing more robust predictions.

In this work, we apply the anchoring augmentation strategy and use an EMA model for computing the pseudo-labels.

3.2.3 Semi-Supervised Semantic Segmentation.

Semi-supervised learning addresses the problem of the high annotation cost by assuming that only a small subset of the available data is labeled, while the rest remains unlabeled. One common approach for this task is to make use of Generative Adversarial Networks (GANs) [80]. Hung et al. [107] propose to train the discriminator to distinguish between confidence maps from labeled and unlabeled data predictions. Mittal et al. [172] make use of a two-branch approach, one branch enforcing low entropy predictions using a GAN approach and another branch for removing false-positive predictions using a Mean Teacher method [231]. A similar idea was proposed by Feng et al. [65], a recent work that introduces Dynamic Mutual Training (DMT). DMT uses two segmentation models and the model’s disagreement is used to re-weight the loss. DMT method also followed the multi-stage training protocol from CBC [64], where pseudo-labels are generated in an offline curriculum fashion. Other works are based on data augmentation methods for consistency regularization. French et al. [67] focus on applying CutOut [57] and CutMix [272], while Olsson et al. [183] propose a data augmentation technique specific for semantic segmentation.

Differently from previous work, we propose a novel feature learning module that shows the benefits of incorporating contrastive learning in a semi-supervised scenario.

3.3 Method

Semi-supervised semantic segmentation consists in a per-pixel classification task where two different sources of data are available: a few fully-labeled samples $\mathcal{X}_l = \{x_l, y_l\}$, where x_l are the training images and y_l their corresponding per-pixel annotations, and a large set of unlabeled samples $\mathcal{X}_u = \{x_u\}$.

To tackle this task, we propose to use a teacher-student scheme. The teacher network f_ξ creates robust pseudo-labels from unlabeled samples and memory bank entries from labeled samples to teach the student network f_θ to improve its segmentation performance.

Teacher-student scheme. The learned weights θ of the student network f_θ are optimized using the following loss function:

$$\mathcal{L} = \lambda_{sup}\mathcal{L}_{sup} + \lambda_{pseudo}\mathcal{L}_{pseudo} + \lambda_{ent}\mathcal{L}_{ent} + \lambda_{contr}\mathcal{L}_{contr}. \quad (3.1)$$

The \mathcal{L}_{sup} is the loss for supervised learning on labeled samples (Section 3.3.1). \mathcal{L}_{pseudo} and \mathcal{L}_{ent} tackle pseudo-labels (Section 3.3.2) and entropy minimization (Section 3.3.3) techniques, respectively, where the pseudo-labels are generated by the teacher segmentation network f_ξ . Finally, \mathcal{L}_{contr} is our proposed contrastive learning loss (Section 3.3.4).

Weights ξ of the teacher network f_ξ are an exponential moving average of weights θ of the student network f_θ with a decay rate $\tau \in [0, 1]$. The teacher model provides

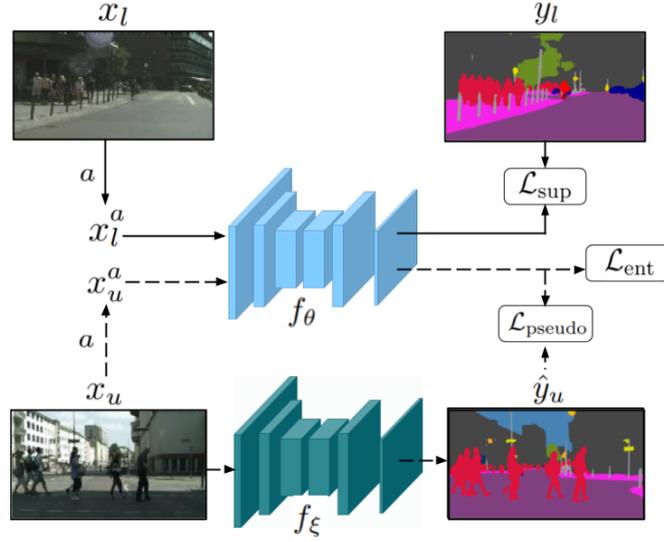


Figure 3.2: **Supervised and self-supervised optimization.** The student network f_θ is optimized by the supervised loss (\mathcal{L}_{sup}) for labeled data (x_l, y_l) . For unlabeled data x_u , the teacher network f_ξ computes the pseudo-labels \hat{y}_u that are later used for optimizing the pseudo-labels loss ($\mathcal{L}_{\text{pseudo}}$) for pairs of augmented samples and pseudo-labels (x_u^a, \hat{y}_u) . Direct entropy minimization (\mathcal{L}_{ent}) is also applied on predictions from x_u^a .

more accurate and robust predictions [231]. Thus, at every training step, the teacher network f_ξ is not optimized by a gradient descent but updated as follows:

$$\xi = \tau\xi + (1 - \tau)\theta. \quad (3.2)$$

3.3.1 Supervised Segmentation: \mathcal{L}_{sup}

Our supervised semantic segmentation optimization, applied to the labeled data \mathcal{X}_l , follows the standard optimization with the weighted cross-entropy loss. Let \mathcal{H} be the weighted cross-entropy loss between two lists of N per-pixel class probability distributions y_1, y_2 :

$$\mathcal{H}(y_1, y_2) = -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C y_2^{(n,c)} \log(y_1^{(n,c)}) \alpha^c \beta^n, \quad (3.3)$$

where C is the number of classes to classify, N is the number of elements, i.e., ixels in y_1 , α^c is a per-class weight, and β^n is a per-pixel weight. Specific values of α^c and β^n are detailed in Section 3.4.2. The supervised loss (see top part of Figure 3.2) is defined as follows:

$$\mathcal{L}_{\text{sup}} = \mathcal{H}(f_\theta(x_l^a), y_l), \quad (3.4)$$

where x_l^a is a weak augmentation of x_l (see Section 3.4.2 for augmentation details).

3.3.2 Learning from Pseudo-labels: $\mathcal{L}_{\text{pseudo}}$

The key to the success of semi-supervised learning is to learn from unlabeled data. One idea our approach exploits is to learn from pseudo-labels. In our case, pseudo-labels

are generated by the teacher network f_ξ (see Figure 3.2). For every unlabeled sample x_u , the pseudo-labels \hat{y}_u are computed following this equation:

$$\hat{y}_u = \arg \max f_\xi(x_u), \quad (3.5)$$

where f_ξ predicts a class probability distribution. Note that pseudo-label generation is performed in an online fashion at each training iteration.

Consistency regularization is introduced by using augmentation anchoring, i.e., computing different data augmentation for each sample x_u on the same batch, which helps the model to converge to a better solution [220]. The pseudo-labels loss for unlabeled data \mathcal{X}_u is calculated by the cross-entropy:

$$\mathcal{L}_{\text{pseudo}} = \frac{1}{A} \sum_{a=1}^A \mathcal{H}(f_\theta(x_u^a), \hat{y}_u), \quad (3.6)$$

where x_u^a is a strong augmentation of x_u and A is the number of augmentations we apply to sample x_u (see Section 3.4.2 for augmentation details).

3.3.3 Direct Entropy Minimization: \mathcal{L}_{ent}

Direct entropy minimization is applied on the class distributions predicted by the student network from unlabeled samples x_u as a regularization loss:

$$\mathcal{L}_{\text{ent}} = -\frac{1}{A} \frac{1}{N} \sum_{a=1}^A \sum_{n=1}^N \sum_{c=1}^C f_\theta(x_u^{a,n,c}) \log f_\theta(x_u^{a,n,c}), \quad (3.7)$$

where C is the number of classes to classify, N is the number of pixels and A is the number of augmentations.

3.3.4 Contrastive Learning: $\mathcal{L}_{\text{contr}}$

Figure 3.3 illustrates our proposed contrastive optimization. A memory bank is filled with high-quality feature vectors from the teacher network f_ξ (right part of Figure 3.3).

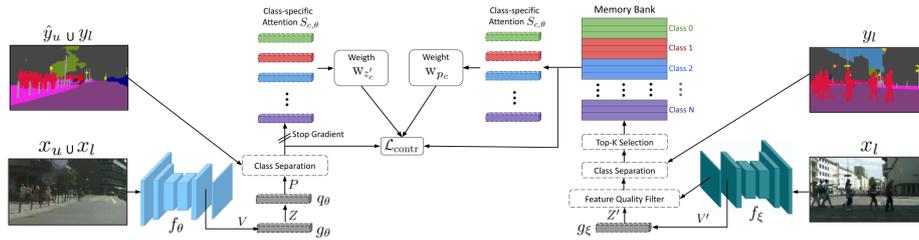


Figure 3.3: **Contrastive learning optimization.** At every iteration, features are extracted by f_ξ from labeled samples (see right part). These features are projected, filtered by their quality, and then, ranked to finally only store the highest-quality features into the memory bank. Concurrently, feature vectors from input samples extracted by f_θ are fed to the projection and prediction heads (see left part). Then, feature vectors are passed to a self-attention module in a class-wise fashion, getting a per-sample weight. Finally, input feature vectors are enforced to be similar to same-class features from the memory bank.

Concurrently, the student network f_θ extracts feature vectors from either \mathcal{X}_l or \mathcal{X}_u . In a per-class fashion, every feature is passed through a simple self-attention module that serves as per-feature weighting in the contrastive loss. Finally, the contrastive loss enforces the weighted feature vectors from the student to be similar to feature vectors from the memory bank. As the memory bank contains high-quality features from all labeled samples, the contrastive loss helps to create a better class separation in the feature space across the whole dataset as well as aligning the unlabeled data distribution with the labeled data distribution.

Contrastive Learning Optimization. Let f_{θ^-} be the student network without the classification layer and $\{x, y\}$ a training sample that is either from the labeled $\{\mathcal{X}_l, Y_l\}$ or unlabeled set $\{\mathcal{X}_u, \hat{Y}_u\}$. The first step is to extract all feature vectors: $V = f_{\theta^-}(x)$. The feature vectors V are then fed to a projection head, $Z = g_\theta(V)$, and a prediction head, $P = q_\theta(Z)$, following [83], where g_θ and q_θ are two different Multi-Layer Perceptrons (MLPs). Next, P is grouped by the different semantic classes in y .

Let $P_c = \{p_c\}$ be the set of prediction vectors from P of a specific class c . Let $Z'_c = \{z'_c\}$ be the set of projection vectors of class c obtained by the teacher network, $Z' = g_\xi(f_{\xi^-}(x))$ from the labeled examples stored in the memory bank, i.e., memory entries.

Next, we learn which feature vectors (p_c and z'_c) are beneficial for the contrastive learning task, by assigning per-feature learned weights (Equation 3.8) that will serve as a weighting factor (Equation 3.10) for the contrastive learning loss function (Equation 3.11). These per-feature weights are computed using class-specific attention modules $S_{c,\theta}$ (see Section 3.4.2 for further details) that generate a single value ($w \in [0, 1]$) for every z'_c and p_c feature. Following [228] we L1 normalize these weights to prevent converging to the trivial all-zeros solution. For the prediction vectors P_c case, the weights w_{p_c} are then computed as follows:

$$w_{p_c} = \frac{|P_c|}{\sum_{p_i \in P_c} S_{c,\theta}(p_i)} S_{c,\theta}(p_c). \quad (3.8)$$

Equation 3.8 is also used to compute $w_{z'_c}$, but using Z'_c and z'_c instead of P_c and p'_c .

The contrastive loss is computed to attract prediction vectors p_c to be similar to projection vectors from the memory bank z'_c . For that, we use the cosine similarity as the similarity measure C :

$$C(p_c, z'_c) = \frac{\langle p_c, z'_c \rangle}{\|p_c\|_2 \cdot \|z'_c\|_2}, \quad (3.9)$$

where, the weighted distance between predictions and memory bank entry is computed by:

$$\mathcal{D}(p_c, z'_c) = w_{p_c} w_{z'_c} (1 - C(p_c, z'_c)), \quad (3.10)$$

and finally, our contrastive loss is computed as follows:

$$\mathcal{L}_{contr} = \frac{1}{C} \frac{1}{|P_c|} \frac{1}{|Z'_c|} \sum_{c=1}^C \sum_{p_c \in P_c} \sum_{z'_c \in Z'_c} \mathcal{D}(p_c, z'_c). \quad (3.11)$$

Memory Bank. The memory bank is the data structure that maintains the target feature vectors z'_c, ψ for each class c , used in the contrastive loss. In our case, it contains only high-quality pixel-level feature vectors from labeled data.

As shown in Figure 3.3, the memory bank is updated on every training iteration with a subset of $z'_c \in Z'$ generated by the teacher network. To select what subset of Z' is included in the memory bank, we first perform a Feature Quality Filter (FQF), where we only keep features that lead to an accurate prediction when the classification layer is applied, $y = \arg \max f_\xi(x_l)$, having confidence higher than a threshold, $f_\xi(x_l) > \phi$. The remaining Z' are grouped by classes Z'_c . Finally, instead of picking randomly a subset of every Z'_c to update the memory bank, we make use of the class-specific attention modules $S_{c,\xi}$. We get ranking scores $R_c = S_{c,\xi}(Z'_c)$ to sort Z'_c and we update the memory bank only with the top-K highest-scoring vectors. The memory bank is a First In First Out (FIFO) queue per class for computation and time efficiency. This way it maintains recent high-quality feature vectors in a very efficient fashion computation-wise and time-wise. Detailed information about the hyper-parameters is included in Section 3.4.2.

3.4 Experiments

This section describes the datasets and implementation details used in the evaluation of the presented work. It also contains the comparison of our method with the state-of-the-art on different benchmarks for semi-supervised semantic segmentation, including a semi-supervised domain adaptation set-up, and a detailed ablation study.

3.4.1 Datasets

- Cityscapes [50]. It is a real urban scene dataset composed of 2975 training and 500 validation samples, with 19 semantic classes.
- PASCAL VOC 2012 [61]. It is a natural scenes dataset with 21 semantic classes. The dataset has 10582 and 1449 images for training and validation respectively.
- GTA5 [200]. It is a synthetic dataset captured from a video game with realistic urban-like scenarios with 24966 images in total. The original dataset provides 33 different categories but, following [252], we only use the 19 classes that are shared with Cityscapes.

3.4.2 Implementation details

Architecture. We use DeepLab networks [34] in our experiments. For the ablation study and most benchmarking experiments, DeepLabv2 with a ResNet-101 backbone is used for a fair comparison (i.e., imilar settings) to previous works [64, 107, 172, 183]. DeepLabv3+ with Resnet50 backbone is also used to equal comparison with [166]. For the teacher network f_ξ , we set $\tau = 0.997$ in (Equation 3.2).

The prediction and projection heads follow [83]: Linear \rightarrow BatchNorm [110] \rightarrow Relu [179] \rightarrow Linear, with a hidden and output dimension of 256. The proposed class-specific attention modules follow a similar architecture: Linear \rightarrow BatchNorm \rightarrow LeakyRelu [157] \rightarrow Linear \rightarrow Sigmoid, with a hidden and output dimension of 256 and 1 respectively. We use $2 \times N_{classes}$ attention modules since they are used in a class-wise fashion. In particular, two modules per class are used because we have different modules for projection or prediction feature vectors.

Following previous works [183, 231], the segmentation is performed with the student network f_θ in the experimental validation.

Optimization. For all experiments, we train for 80K iterations using the SGD optimizer with a momentum of 0.9. The learning rate is set to 2×10^{-4} for DeepLabv2 and 4×10^{-4} for DeepLabv3+ with a poly learning rate schedule. For the Cityscapes and GTA5 datasets, we use a crop size of 512×512 and batch sizes of 5 and 7 for Deeplabv2 and Deeplabv3+, respectively. For Pascal VOC, we use a crop size of 321×321 and batch sizes of 14 and 20 for Deeplabv2 and Deeplabv3+, respectively. Cityscapes images are downsampled to 512×1024 before cropping when Deeplabv2 is used for a fair comparison with previous works [64, 107, 172, 183]. The different loss weights in (Equation 3.1) are set as follows for all experiments: $\lambda_{sup} = 1$, $\lambda_{pseudo} = 1$, $\lambda_{ent} = 0.01$, $\lambda_{contr} = 0.1$. An exception is made for the first 2K training iterations where $\lambda_{contr} = 0$ and $\lambda_{pseudo} = 0$ to make sure predictions have some quality before being taken into account. Regarding the per-pixel weights (β^n) from \mathcal{H} in (Equation 3.3), we set it to 1 for \mathcal{L}_{sup} . For \mathcal{L}_{pseudo} , we follow [64] weighting each pixel with its corresponding pseudo-label confidence with a sharpening operation, $f_\xi(x_u)^s$, where we set $s = 6$. As for the per-class weights α^c in (Equation 3.3), we perform a class balancing for the Cityscapes and GTA5 datasets by setting $\alpha^c = \sqrt{\frac{f_m}{f_c}}$ with f_c being the frequency of class c and f_m the median of all class frequencies. In semi-supervised settings the amount of labels, Y_l , is usually small. For a more meaningful estimation, we compute these frequencies not only from Y_l but also from \hat{Y}_u . For the Pascal VOC we set $\alpha^c = 1$ as the class balancing does not have a beneficial effect.

Other details. DeepLab’s output resolution is $\times 8$ lower than the input resolution. For feature comparison during training, we keep the output resolution and downsample the labels reducing memory requirements and computation.

The memory bank size is fixed to $\psi = 256$ vectors per class (see Section 3.4.4 for more details). The confidence threshold ϕ for accepting features is set to 0.95. The number of vectors added to the memory bank at each iteration, for each image, and for each class is set as $\max(1, \frac{\psi}{|\mathcal{X}_l|})$, where $|\mathcal{X}_l|$ is the number of labeled samples.

A single NVIDIA Tesla V100 GPU is used for all experiments. All our reported results are the mean of three different runs with different labeled/unlabeled data splits.

Data augmentation. In our work, we use two different augmentation set-ups, a weak augmentation for labeled samples and a strong augmentation set-up for unlabeled samples, see Table 3.1 for configuration details. For the augmentation anchoring (Equation 3.6), we set $A = 2$ as the number of augmentations for the same sample.

Table 3.1: Strong and weak data augmentation set-ups

Parameter	Weak	Strong
Flip probability	0.50	0.50
Resize $\times [0.75, 1.75]$ probability	0.50	0.80
Color jittering probability	0.20	0.80
Brightness adjustment max intensity	0.15	0.30
Contrast adjustment max intensity	0.15	0.30
Saturation adjustment max intensity	0.075	0.15
Hue adjustment max intensity	0.05	0.10
Gaussian blurring probability	0	0.20
ClassMix [183] probability	0.20	0.80

3.4.3 Benchmark Experiments

The following experiments compare the proposed method with state-of-the-art methods in different semi-supervised semantic segmentation set-ups, including the semi-supervised domain adaptation scenario.

Semi-supervised Semantic Segmentation

Cityscapes. Table 3.2 compares different methods on the Cityscapes benchmark for different labeled-unlabeled rates: $\frac{1}{30}$, $\frac{1}{8}$ and $\frac{1}{4}$. Fully Supervised (FS) scenario, where all images are labeled, is also shown as a reference. As shown in the table, our approach outperforms the current state-of-the-art by a significant margin in all settings. The

Table 3.2: Performance (Mean IoU) for the Cityscapes *val* set for different labeled-unlabeled ratios and, in parentheses, the difference w.r.t. the corresponding fully supervised (FS) result.

method	1/30	1/8	1/4	FS
<i>Architecture: Deeplabv2 with ResNet-101 backbone</i>				
Adversarial [107]+	—	58.8 (-7.6)	62.3 (-4.1)	66.4
s4GAN [172]*	—	59.3 (-6.7)	61.9 (-4.9)	66.0
French et al. [67]*	51.2 (-16.3)	60.3 (-7.2)	63.9 (-3.6)	67.5
CBC [64]+	48.7 (-18.2)	60.5 (-6.4)	64.4 (-2.5)	66.9
ClassMix [183]+	54.1 (-12.1)	61.4 (-4.8)	63.6 (-2.6)	66.2
DMT [65]*+	54.8 (-13.4)	63.0 (-5.2)	—	68.2
Ours*	58.0 (-8.4)	63.0 (-3.4)	64.8 (-1.6)	66.4
Ours+	59.4 (-7.9)	64.4 (-2.9)	65.9 (-1.4)	67.3
<i>Architecture: Deeplabv3+ with ResNet-50 backbone</i>				
Error-corr [166]*	—	67.4 (-7.4)	70.7 (-4.1)	74.8
Ours*	64.9 (-9.3)	70.0 (-4.2)	71.6 (-2.6)	74.2

* *ImageNet pre-training, + COCO pre-training*

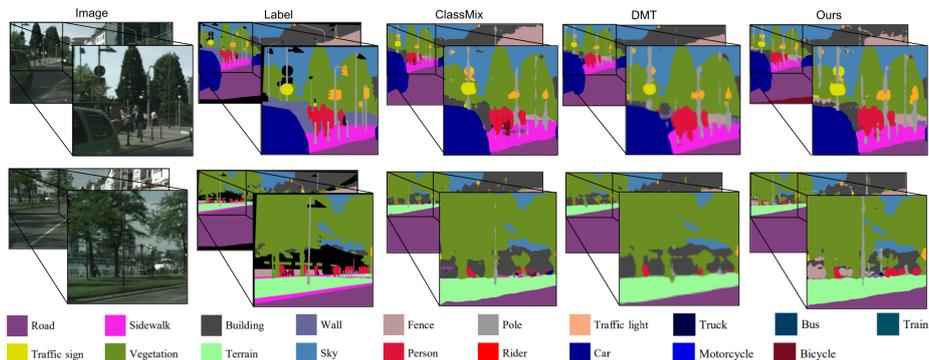


Figure 3.4: **Qualitative results on Cityscapes.** Models are trained with $\frac{1}{8}$ of the labeled data using Deeplabv2 with ResNet-101. From left to right: Image, manual annotations, ClassMix [183], DMT [65], our approach.

performance difference is increasing as less labeled data is available, demonstrating the effectiveness of our approach. This is particularly important since the goal of semi-supervised learning is to learn with as little supervision as possible. Note that the upper bound for each method is shown in the fully supervised setting (FS).

Figure 3.4 shows a visual comparison of the top-performing methods on different relevant samples from Cityscapes. Note in these examples how our approach improves on fine-grained examples (e.g., oles, traffic lights, signs, or people).

Pascal VOC. Table 3.3 shows the comparison of different methods on the Pascal VOC benchmark, using different labeled-unlabeled rates: $\frac{1}{50}$, $\frac{1}{20}$ and, $\frac{1}{8}$. Our proposed method outperforms previous methods for most of the configurations. Like in the previous benchmark, our method presents larger benefits for the more challenging cases, i.e., nly a small fraction of data is labeled ($\frac{1}{50}$). This demonstrates that the proposed approach is especially effective to learn from unlabeled data.

Semi-supervised Domain Adaptation Semi-supervised domain adaptation for semantic segmentation differs from the semi-supervised set-up in the availability of labeled data from another domain. That is, apart from having $\mathcal{X}_l = \{x_l, y_l\}$ and $\mathcal{X}_u = \{x_u\}$ from the target domain, a large set of labeled data from another domain is also available: $\mathcal{X}_d = \{x_d, y_d\}$.

Our method can naturally tackle this task by evenly sampling from both \mathcal{X}_l and \mathcal{X}_d as our labeled data when optimizing \mathcal{L}_{sup} and $\mathcal{L}_{\text{contr}}$. However, the memory bank only stores features from the target domain \mathcal{X}_l . In this way, both the features from unlabeled data \mathcal{X}_u , and the features from the other domain \mathcal{X}_d are aligned with those from \mathcal{X}_l .

Following ASS [252], we take the GTA5 dataset as \mathcal{X}_d , where all elements are labeled, and the Cityscapes is the target domain consisting of a small set of labeled data \mathcal{X}_l and a large set of unlabeled samples \mathcal{X}_u . Table 3.4 compares the results of our method with ASS, state-of-the-art on this task, both using ImageNet pre-training

Table 3.3: Performance (Mean IoU) for the Pascal VOC *val* set for different labeled-unlabeled ratios and, in parentheses, the difference w.r.t. the corresponding fully supervised (FS) result.

method	1/50	1/20	1/8	FS
<i>Architecture: Deeplabv2 with ResNet-101 backbone</i>				
Adversarial [107]+	57.2 (-17.7)	64.7 (-10.2)	69.5 (-5.4)	74.9
s4GAN [172]+	63.3 (-10.3)	67.2 (-6.4)	71.4 (-2.2)	73.6
French et al. [67]*	64.8 (-7.7)	66.5 (-6.0)	67.6 (-4.9)	72.5
CBC [64]+	65.5 (-8.1)	69.3 (-4.3)	70.7 (-2.9)	73.6
ClassMix [183]+	66.2 (-7.9)	67.8 (-6.3)	71.0 (-3.1)	74.1
DMT [65]*+	67.2 (-7.6)	69.9 (-4.9)	72.7 (-2.1)	74.8
Ours*	65.4 (-7.2)	67.8 (-5.1)	69.9 (-2.7)	72.6
Ours+	67.9 (-6.2)	70.0 (-4.1)	71.6 (-2.5)	74.1
<i>Architecture: Deeplabv3+ with ResNet-50 backbone</i>				
Error-corr [166]*	—	—	70.2 (-6.1)	76.3
Ours*	63.4 (-12.5)	69.1 (-6.8)	71.8 (-4.1)	75.9

* ImageNet pre-training, + COCO pre-training

Table 3.4: Mean IoU in Cityscapes *val* set. Central columns evaluate the semi-supervised domain adaptation task (GTA5 \rightarrow Cityscapes). The last column evaluates a semi-supervised setting in Cityscapes (no adaptation). Different labeled-unlabeled ratios for Cityscapes are compared. All methods use ImageNet pre-trained Deeplabv2 with ResNet-101 backbone.

City Labels	ASS [252] with domain adaptation	Ours	Ours no adaptation
1/30	54.2	59.9	58.0
1/15	56.0	62.0	59.9
1/6	60.2	64.2	63.7
1/3	64.5	65.6	65.1

Table 3.5: Ablation study on the different losses included (Equation 3.1). Mean IoU obtained on Cityscapes benchmark ($\frac{1}{30}$ available labels, Deeplabv2-ResNet101 COCO pre-trained).

\mathcal{L}_{sup}	$\mathcal{L}_{\text{pseudo}}$	\mathcal{L}_{ent}	$\mathcal{L}_{\text{contr}}$	mIoU
✓				49.5
✓	✓			56.7
✓		✓		52.2
✓			✓	54.4
✓	✓	✓		57.4
✓	✓		✓	59.0
✓		✓	✓	57.3
✓	✓	✓	✓	59.4

for a fair comparison. For reference, we also show the results of our approach with no adaptation, i.e., only training on the target domain Cityscapes, as we do for the semi-supervised set up from the previous experiment (Table 3.2). We can see that our approach benefits from the use of the other domain data (GTA5), especially where there is little labeled data available ($\frac{1}{30}$), as it could be expected. Our method outperforms ASS by a large margin in all the different set-ups. As in previous experiments, our improvement is more significant when the amount of available labeled data is smaller.

3.4.4 Ablation Experiments

The following experiments study the impact of the different components of the proposed approach. The evaluation is done on the Cityscapes data, since it provides more complex scenes compared to Pascal VOC. We select the challenging labeled data ratio of $\frac{1}{30}$.

Losses impact. Table 3.5 shows the impact of every loss used by the proposed method. We can observe that the four losses are complementary, getting a 10 mIoU increase over our baseline model, using only the supervised training when $\frac{1}{30}$ of the Cityscapes labeled data is available. Note that our proposed contrastive module $\mathcal{L}_{\text{contr}}$ is able to get

Table 3.6: Effect of different values for the factor λ_{contr} (Equation 3.1) that weights the effect of the contrastive loss \mathcal{L}_{contr} . Results on Cityscapes benchmark ($\frac{1}{30}$ available labels, Deeplabv2-ResNet101 COCO pre-trained).

λ_{contr}	10^4	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-4}
mIoU	50.3	51.4	54.8	59.1	59.4	58.7	57.6

Table 3.7: Effect of our contrastive learning memory bank size (features per-class), ψ . Results on Cityscapes benchmark ($\frac{1}{30}$ available labels, Deeplabv2-ResNet101 COCO pre-trained).

ψ	32	64	128	256	512
mIoU	58.7	58.9	59.2	59.4	59.3

54.32 mIoU even without any other complementary loss, which is the previous state-of-the-art for this set-up (see Table 3.2). Adding the \mathcal{L}_{pseudo} significantly improves the performance and then, adding \mathcal{L}_{ent} regularization loss gives a little extra performance gain.

Contrastive module. Table 3.6 shows an ablation on the influence of the contrastive module for different values of λ_{contr} (Equation 3.1). As expected, if this value is too low, the effect gets diluted, with similar performance as if the proposed loss is not used at all (see Table 3.5). High values are also detrimental, probably because it acts as increasing the learning rate vastly, which hinders the optimization. The best performance is achieved when this contrastive loss acts as auxiliary ($\lambda_{contr} = 10^{-1}$), i.e., its weight is a little lower than the segmentation losses (\mathcal{L}_{sup} and \mathcal{L}_{pseudo}).

The effect of the size (per-class) of our memory bank is studied in Table 3.7. As expected, higher values lead to stronger performances, although from 256 up they tend to maintain similarly. Because all the elements from the memory bank are used during

Table 3.8: Ablation study of our contrastive module main components. Results on Cityscapes benchmark ($\frac{1}{30}$ available labels, using Deeplabv2-ResNet101 COCO pre-trained).

Base	f_ξ	$S_{c,\theta}$	FQF	mIoU
✓				58.3
✓	✓			58.7
✓		✓		58.6
✓			✓	59.0
✓	✓	✓	✓	59.4

f_ξ : Use teacher model f_ξ to extract features instead of f_θ
 $S_{c,\theta}$: Use class-specific attention $S_{c,\theta}$ to weight every feature
 FQF: Feature Quality Filter for Memory Bank update

the contrastive optimization (Equation 3.11) the larger the memory bank is, the more computation and memory it requires. Therefore, we select 256 as our default value.

Table 3.8 studies the effect of the main components used in the proposed contrastive learning module. The base configuration of the module still presents a performance gain compared to not using the contrastive module. This shows that our simplest implementation of the per-pixel contrastive learning using the memory bank already helps to improve the segmentation task in a semi-supervised scenario. Generating and selecting good quality prototypes is the most important factor. This is done both by the Feature Quality Filter (FQF), i.e., heeding that the feature leads to an accurate and confident prediction, and extracting them with the teacher network f_ξ . Then, using the class-specific attention $S_{c,\theta}$ to weight every sample (both from the memory bank and input sample) is also beneficial, acting as a learned sampling method.

3.5 Conclusion

This work presents a novel approach for semi-supervised semantic segmentation. Our work shows the benefits of incorporating contrastive learning techniques to solve this semi-supervised task. The proposed contrastive learning module boosts the performance of semantic segmentation in these settings. Our new module contains a memory bank that is continuously updated with selected features from those produced by a teacher network from labeled data. These features are selected based on their quality and relevance for the contrastive learning. Our student network is optimized for both labeled and unlabeled data to learn similar class-wise features to those in the memory bank. The use of contrastive learning at a pixel-level has been barely exploited and this work demonstrates the potential and benefits it brings to semi-supervised semantic segmentation and semi-supervised domain adaptation. Our results outperform state-of-the-art on several public benchmarks, with particularly significant improvements on the more challenging set-ups, i.e., when the amount of available labeled data is low.

Chapter 4

Efficient Semantic Segmentation

This chapter tackles the very relevant challenge of building efficient models. Efficiency is another important factor to consider when aiming to apply semantic segmentation solutions in real-world settings. Frequently, state-of-the-art CNN models, especially for semantic segmentation, require high-end GPUs to run inferences in near real-time. However, high-end GPUs are not always available, for example in many robotic platforms or for mobile phone applications. This chapter discusses our work towards solutions to this problem. First, it introduces the proposed memory-efficient and fast semantic segmentation model (MiniNet) able to run in CPU at acceptable frame-rates and consuming low computation resources. In order to show the applicability of this CPU segmentation model, the model is applied as a key element to the problem of visual SLAM keyframe selection. Secondly, Mininet-v2 is described, that further improves the proposed efficient model for CPU and proposes a novel architecture for low-end GPUs.

4.1 Efficient Semantic Segmentation to enhance V-SLAM Keyframe Selection

Visual SLAM is an essential task running in the back-end of many robotic systems, but the mapping itself is often not the final goal on the robot missions. In recent years, it is more and more common to have robots or teams of robots communicating with a central station or the rest of the team members to achieve more sophisticated or high-level tasks.

Our work explores the possibilities of selecting more carefully the keyframes that the V-SLAM uses for mapping and place recognition, in order to be able to re-use them for additional posterior tasks. This way, we enable more efficient use of those keyframes that need to be stored and probably transmitted. Currently, additional data would need to be used for those posterior tasks, such as recognition of objects or elements of interest in the mapped environment.

State-of-the-art approaches for visual recognition tasks have witnessed a significant boost and outstanding performances lately thanks, among other reasons, to deep learning based solutions. There have been only a few years since Convolutional Neural Networks (CNNs) caught significant attention [130] and have already been adopted for numerous commercial products. Although CNN models inference time is very short compared to the training time, it is usually required to have high-end GPU(s) to run in-

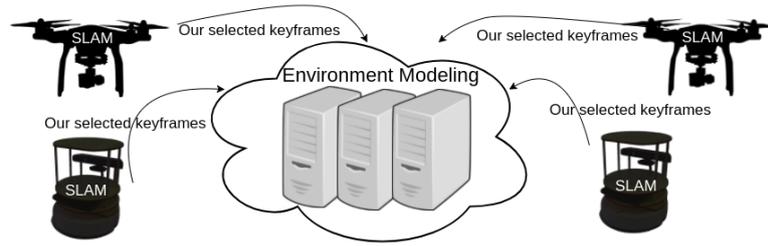


Figure 4.1: **Our approach runs a smart keyframe selection at each robot on-board CPU.** Selected keyframes are stored and/or shared across the system, typically for further more complex processing.

ferences in near real-time. Unfortunately, these GPUs are often not available in robotic platforms, which present restrictions incompatible with the use of high-end GPUs, such as small robots or drones that cannot hold the extra weight or afford the extra power consumption.

That is why in the last years, we see an increasing interest on Deep Learning solutions for real-time applications on low-power GPUs [97, 187, 202]. They get results close to the state-of-the-art at much lower computational and energy cost, as discussed in more detail later. Nevertheless, even these architectures cannot run on CPUs with the required execution times, although still CPU is the only computing source available for many robots.

This work presents a novel and efficient strategy to include additional criteria to commonly used V-SLAM keyframe selection. Our contribution is twofold:

- A novel strategy for more meaningful keyframe selection, while a robot is mapping its environment, which runs efficiently on the robot CPU.
- A new CNN architecture for semantic segmentation (MiniNet) developed to be able to run on the robot CPU and serve as quick semantic filtering of frames.

Our approach obtains more representative keyframes with little extra cost and, by re-using the keyframes for multiple tasks, avoids extra computations or communications. This is particularly relevant in multi-robot settings where computation and communications bottlenecks are critical. Multi-robot teams often have several nodes with heterogeneous computational capabilities, as illustrated in Fig. 4.1. They present scenarios where efficiently selecting the most representative information is important to minimize the amount of information shared. For example, the well-known DARPA Subterranean (SubT) Challenge¹, presents a real use-case where communication restrictions are very strong, therefore selecting carefully what to transmit is critical.

The proposed MiniNet gets comparable results to state-of-the-art on segmentation tasks with few classes. It can run onboard the robot CPU to perform tasks such as the presented semantic based keyframe selection, but other applications could benefit from this architecture. The keyframe selection proposed is shown to pick more representative information for high-level recognition tasks (text reading), without losing more basic navigation information (relevant information for place recognition).

¹<https://www.subtchallenge.com/>

4.1.1 Related Work

The most relevant literature to our contributions is related to selecting relevant keyframes within sequences and to efficient convolutional neural network architectures.

Keyframe selection Selecting the most representative and valuable frames out of a sequence is, in fact, a visual summarization problem. Depending on the problem and scenario, the criteria and the meaning of valuable information differs. For general video summarization, the selection targets the most representative frames which condense all the events of the entire video. As for many other tasks, deep learning based approaches are leading current state of the art, such as applying recurrent methods [275], CNNs for ranking methods [267] or semantic embeddings [271] for summarization tasks.

For more specific applications, such as surveillance, more specific contents need to be selected, and additional restrictions, such as computational resource or execution time, need to be considered [153]. This type of approaches is closer to our goals since these restrictions also affect robotic applications. Mobile robots need to perform several real-time tasks in parallel (e.g., V-SLAM or visual recognition algorithms), and cannot afford to apply heavy techniques such as the ones used in general video summarization. Well-known VSLAM algorithms, such as ORBSLAM2 [178], need to select keyframes to reduce the data used for tracking and place recognition tasks. When modeling the environment with multi-robot teams [199, 212], these keyframes become also the information shared among the robots but still follow the standard VSLAM selection criteria, even though other nodes with higher capabilities could perform more demanding tasks if we would select more carefully what to share.

CNN architectures for low computational environments Many works lately focus on reducing CNNs memory and computational cost, which directly affects energy consumption and inference time. Some approaches focus on the training phase (e.g., joint training and distillation [93, 193, 204]), others on parameters data type (e.g., quantized [103] or binary [51] networks) or post-processing methods (e.g., pruning [88, 174]) and others on novel architecture operations (e.g. depth-wise separable convolutions [116], dilated convolutions [270] and self normalizing neural networks [125]).

Our work is focused on running efficiently semantic segmentation tasks. CNNs for semantic segmentation typically follow an encoder-decoder structure: an encoder which learns features while reducing the resolution and a decoder which upsamples the learned features and maps them into the segmentation result. Recent works towards efficient segmentation architectures include Deeplab-v3 [35, 37] and ERFNet [202], that use atrous convolutions [270] to avoid the need to reduce much the input resolution. Many architectures targeting efficiency, e.g., ERFNet [202] and ENet [187], perform several consecutive early downsampling operations for a quick reduction of the input resolution and they have light decoders with very few parameters and layers.

Our proposed architecture is inspired by many of these recent works, focused on efficient semantic segmentation tasks but, differently from other works, considering the feasibility of CPU execution. We focus on semantic segmentation because it provides information about the whole image scene (pixel-level semantic labels), essential to have a quick frame content analysis.

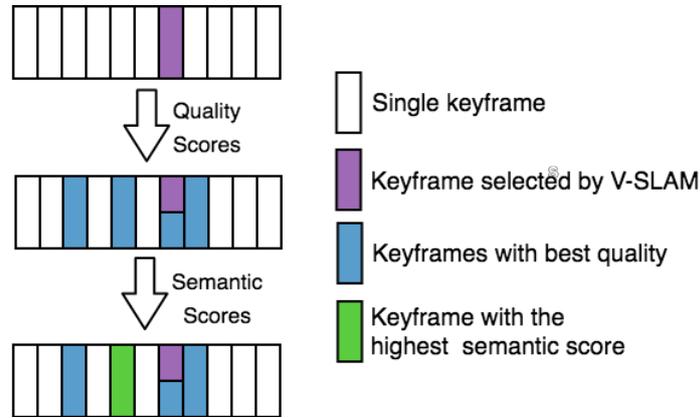


Figure 4.2: **Keyframe selection proposed.** (Top) Define a window of keyframe candidates around each selected keyframe by VSLAM; (middle) compute the quality score to reduce the set; (bottom) pick the top best quality frame according to our semantic score. Best viewed in color.

4.1.2 Efficient semantic based keyframe selection on the robot CPU.

Our proposed approach to select the most relevant keyframes has been designed with two requirements in mind: 1) A versatile framework to combine several scoring functions about the *relevance* of the keyframes for both local robot operations and global team goals. 2) Run at acceptable rates locally on the robot CPU, i.e., around 20-30 fps.

To account for the first requirement, we propose a hierarchical scoring system to select the most relevant keyframes processed during mapping (summarized in Fig. 4.2). To account for the limited computational resources, we introduce a novel CNN architecture, *MiniNet*, that enables a rough but very fast semantic segmentation on CPU.

Keyframe selection algorithm The core idea of our keyframe selection algorithm is to boost typical VSLAM criteria (select a new keyframe when the established geometric change is reached) by selecting a higher quality and meaningful keyframes for posterior place recognition of other robots, relocalization, and further visual analysis tasks. As Fig. 4.2 illustrates, our system runs an evaluation on a window around each of the original VSLAM keyframes and selects the overall best combining two criterion:

Image quality criterion we first set a candidate window around each keyframe selected by the VSLAM. We define the *quality* of an image I as a combination of two scores, defined in eq. (4.1). The first score, blurriness score sc_{BL} defined in eq. (4.2), is based on the Energy of Laplacian [189], where ∂I is the Laplacian of I . The higher the value of this score, the more likely to be selected. The second score, brightness score sc_{BR} defined in eq. (4.3), is based on the total luminance on the image pixels. The image I is converted to LAB colorspace and the L channel values are zero-centered and added. The higher this score, the lower the image quality. To keep the computational cost low, we use a 112×112 resolution. Before combining the two scores, we independently normalize each one dividing by the corresponding maximum value in

each candidates window.

$$sc_{quality}(I) = norm(sc_{BL}(I)) * norm(sc_{BR}(I)) \quad (4.1)$$

$$sc_{BL}(I) = \sum_{(i,j) \in \Omega(x,y)} \partial I(i,j)^2 \quad (4.2)$$

$$sc_{BR}(I) = \frac{1}{\sum_{(i,j) \in \Omega(x,y)} zero_center(L(i,j))^2} \quad (4.3)$$

Semantic content criterion the second part of the selection algorithm focuses on the image semantic content which may be relevant for the high-level tasks to be performed. This step only evaluates the Q frames with higher $sc_{quality}$ score and computes the semantic score for each of them. As a concrete use case to demonstrate this step, let us think of a system focused on finding textual information in the environment. However, note that the proposed *MiniNet* for quick semantic filtering, detailed in next subsection 4.1.3, can be fine-tuned for different target semantic classes.

The proposed semantic score is based on a rough semantic segmentation, achieved efficiently by the proposed *MiniNet*. This score, eq. (4.4), is computed as the ratio of image pixels that belong to the target class, penalizing the ratio of pixels from the target class which lay on the image border. This penalizes images where the target objects are very likely to be only partially visible, e.g., if a text region is next to the border, it is likely to have only half of a sign visible:

$$sc_{semantic}(I) = \frac{\sum_{(i,j) \in \Omega(x,y)} Text(i,j)}{1 + (\sum_{(i,j) \in \Omega(x,y)} Text_{border}(i,j))}, \quad (4.4)$$

where $Text(i,j)$ is the text segmentation value of image pixel i, j (1 for text pixels, 0 otherwise). Same values for $Text_{border}(i,j)$, text on image borders.

4.1.3 *MiniNet* network architecture

The proposed architecture for semantic segmentation² is designed to efficiently run on CPU, which increases the applicability of CNNs for robotic tasks with execution time restrictions. In this work, *MiniNet* is used to build the $sc_{semantic}$ score, eq. (4.4). However, we should note that it could be beneficial on its own for many other visual tasks run on restricted robotic platforms, independently of the use of a VSLAM algorithm or not. *MiniNet* architecture is inspired by several prior works on CNNs for low computation environments, as discussed in Sec. 4.1.1, in particular ERFNet [202] and ENet [187], with the particularity that our work takes into account the best options for execution on CPU.

The *MiniNet* blocks (detailed in Figure 4.3 and Table 4.1) are the following:

Downsampling block the purpose of this block is to reduce the resolution to a reasonable one on which to perform thorough feature extraction. The input resolution is 512x256 which is a reasonable input size compared to the state-of-the-art [37,112,187]. Our downsample operation performs a depth-wise separable convolution with a stride of 2x2. The proposed architecture performs four downsample operations, leading to a 32x16 resolution.

²Link to the official available implementation: <https://github.com/Shathe/MiniNet>.

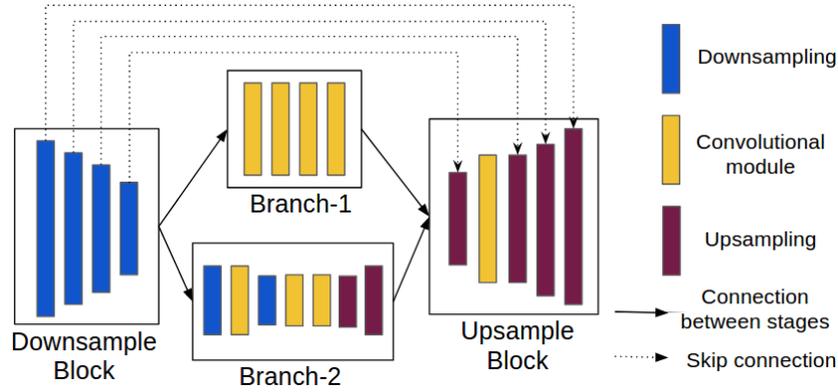


Figure 4.3: **MiniNet architecture diagram.** See Table 4.1 for further detail.

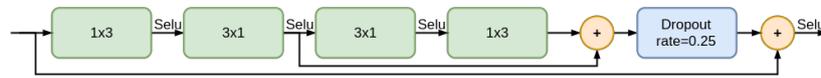


Figure 4.4: **Convolutional module:** Four separable convolutions with two residual connections. Lastly a dropout layer is applied to help dealing with overfitting.

Table 4.1: *MiniNet* Architecture. It has four main blocks: downsample, two convolutional branches and upsample.

	Name	Type	Input	Output size
Downsample	d1	downsampling	image	256x128x12
	d2	downsampling	d1	128x64x24
	d3	downsampling	d2	64x32x48
	d4	downsampling	d3	32x16x96
Branch-1	branch_1.1	module rate=1	d4	32x16x96
	branch_1.2	module rate=2	branch_1.1	32x16x96
	branch_1.3	module rate=4	branch_1.2	32x16x96
	branch_1.4	module rate=8	branch_1.3	32x16x96
Branch-2	d5	downsampling	d4	16x8x192
	branch_2.1	module rate=1	d5	16x8x192
	d6	downsampling	branch_2.1	8x4x386
	branch_2.2	module rate=1	d6	8x4x386
	branch_2.3	module rate=1	branch_2.2	8x4x386
	up1	upsampling	branch_2.3	16x8x192
	branch_2.4	module rate=1	up1	16x8x192
	up2	upsampling	branch_2.4, d5	32x16x96
Upsample	up3	upsampling	branch_1.4, up2, d4	64x32x96
	module_up	module rate=1	up3	64x32x48
	up4	upsampling	module_up, d3	128x64x24
	up5	upsampling	up4, d2	256x128x12
	output	upsampling	up5, d1	512x256xN

Two convolutional branches the network is split into two parallel convolutional branches. These branches use our *convolutional module* (see Fig. 4.4). This mod-

ule is based on the ERFNet Non-bottleneck-1D module [202]. Our main modifications are:

- Include a sum operation between the two decomposed convolutions to conserve the output of the intermediate convolution.
- Remove the Batch Normalization [110] (and Relu [180]) and adding in stead Selu activations, i.e., self normalizing neural networks [125], gaining a $\times 2$ of speed in CPU.
- Change standard convolutions for separable convolutions. Performing depth-wise separable instead of standard convolutions reduces around 2-3 times the computation [97].
- Instead of using the standard dropout, we perform the alpha dropout [125].

The *branch-1 block* consists of four consecutive *convolutional modules* with different dilatation rates. This branch performs parameter-efficient feature extraction on a higher resolution (32x16) than the other branch thanks to dilated convolutions. We cannot afford to add more than four convolutional modules at this resolution for real-time performance on CPU. This fact shows the differences between CPU and low-powered GPUs, where 10-20 modules of this type can be processed and even at higher resolutions.

The *branch-2 block* follows the regular encoder architecture with no-dilated convolutions working on a tiny resolution. This branch plays a very important role in this architecture allowing more time-efficient feature extraction. This branch consists of applying several downsampling and convolutional modules up to 8x4 resolution performing the feature extraction and then, upsampled the features up to the initial size (32x16).

Decoder block our upsample operation consists of a transposed convolution (kernel 3x3 and stride of 2x2). The two convolutional branches are concatenated, upsampled and applied a convolutional module. Then, we concatenate the features with skip connections from the downsample part and perform an upsample operation. This is repeated until getting the initial 512x256 resolution.

4.1.4 Experiments

This section validates the effectiveness of the keyframe selection algorithm presented and evaluates the performance of the proposed *MiniNet*.

MiniNet performance and suitability The following two experiments compare its performance to state-of-the-art CNNs on common segmentation benchmarks, detailed later in each experiment. The first one, Cityscapes, is a more general multi-class segmentation benchmark, in order to have a general evaluation of the reach of *MiniNet*. The second one, COCO-Text, is a more specific binary-segmentation benchmark, to evaluate the network on the more specific type of tasks expected to be part of the keyframe selection proposed.

Training details *MiniNet* has been trained for 90K iterations on the Cityscapes dataset and for 20K iterations on the COCO-Text dataset using a batch size of 32. We use Adam optimizer [123] with initial learning rate of $n = 10^{-3}$ and polynomial learning rate decay. We optimize it through the cross-entropy loss function commonly used to train segmentation models. To account for class imbalance, we use the median frequency class balancing, as applied in SegNet [16]. To smooth the resulting class

Table 4.2: Segmentation results on CityScapes online benchmark

CityScapes (19-classes) Metrics*				
	<i>Cla-IoU</i>	<i>Cat-IoU</i>	GPU (s)	CPU (s)
DeeplabV3+ [37]	82.1	92.0	0.512	14.392
ERFNet [202]	69.7	87.3	0.024	0.571
ENet [187]	58.3	80.4	0.013	n/a
RTSeg [216]	58.3	80.2	n/a	n/a
MiniNet (ours)	40.7	70.5	0.004	0.018

**Cla* = Class; *Cat* = Category; IoU = Intersection over Union metric

**GPU* = forward pass time on Titan X; *CPU* = forward pass time on Intel i5-8600k

weights, we propose to apply a power operation: $w_c = \left(\frac{f_{median}}{f_c}\right)^i$, with f_c being the frequency of class c and f_{median} the median of all frequencies.

The image augmentation applied in all experiments consists of left-right flips, small spatial shifts and scales (up to 10%) and small contrast normalization (α between 0.90 and 1.20). Additionally, for the binary segmentation experiment, we include *black-and-white* augmentation, i.e., randomly converting the RGB image into a grayscale one. This helps to generalize better to gray-scale test images (very common in robotics).

Multi-class segmentation experiment This first experiment compares the state-of-the-art with *MiniNet* results on the Cityscapes dataset [50], an urban scene dataset commonly used to evaluate semantic segmentation approaches. This evaluation is done automatically on the dataset official benchmark site by submitting the test predictions. Table 4.2 shows the performance of our approach, using the public benchmark metrics, for the most relevant methods to our work published on that site: Deeplabv3+ is currently the overall state-of-the-art, while ERFNet and ENet are the best on low-power GPUs considering the trade-off between performance and speed. If available, we report the execution times for GPU. For CPU times, we have computed the mean of 5 executions (ran with the authors’ available code). Note that we were not able to run ENet on CPU due to the lack of CPU implementation of the required operation *MaxPooling with argmax*. *MiniNet* is able to run $\times 3$ faster than ENet on GPU, but gets 18% less Cla-IoU and 10% less Cat-IoU. Thus, as far as GPU is concerned, the trade-off between speedup and loss of IoU seems proportional in both cases. However, concerning the CPU time performance, note that *MiniNet* is over $\times 30$ times faster than ERFNet (while in GPU is $\times 6$ times faster).

These results confirm the proposed architecture gets reasonable accuracy in general tasks while achieving much faster execution, especially in CPU, which is particularly important for our goals: quickly filtering images on each robot to select what’s worth sharing for further processing.

Binary segmentation experiment This experiment focuses on *text segmentation*, which may seem an easier task than the previous experiment, but it is closer to the type of quick filtering tasks that *MiniNet* is designed to work with. As we analyze in the next section, a use case of the keyframe selection strategy proposed in this work is to quickly filter keyframes on the robot where text regions seem more significant to facilitate further text reading tasks on the selected frames. For this experiment, we use the well known COCO-Text dataset [243] (a subset of machine printed and legible text

Table 4.3: Segmentation results on COCO-Text

	Text Segmentation (binary) Metrics*					
	GPU(s)	CPU(s)	GFlops	R	P	IoU
DeeplabV3+ [37]	0.512	14.392	102.85	58.85	43.90	33.29
ERFNet [202]	0.024	0.571	55.21	52.66	39.73	29.27
MiniNet (ours)	0.004	0.018	1.06	52.61	36.69	27.63

* R = Recall; P = Precision; IoU = Intersection over Union metric

*GPU = forward pass time on Titan X; CPU = forward pass time on Intel i5-8600k

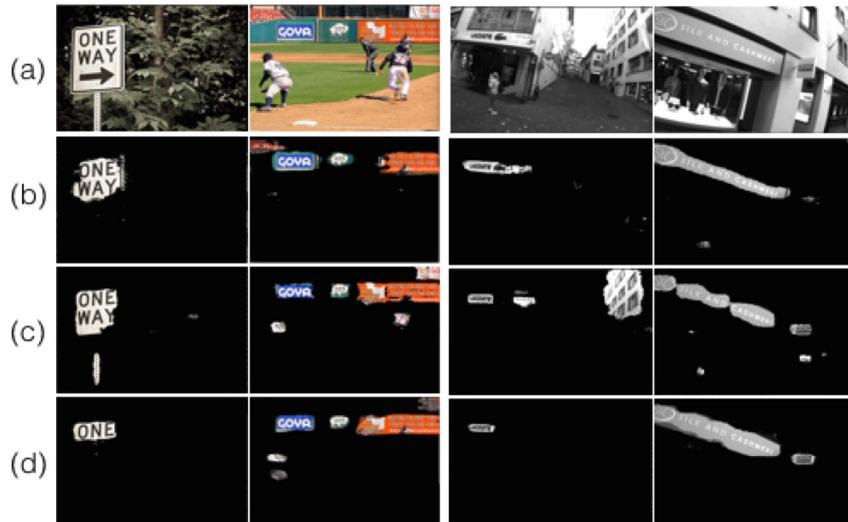


Figure 4.5: Segmentation from COCO-Text (left) and V4RL data (right). (a) input image, (b) MiniNet, (c) Deeplabv3+ and (d) ERFNet segmentations.

images). Text in this dataset is labeled for text detection with bounding boxes, but we use them as approximated per-pixel annotations for our segmentation results. For this experiment, we trained from scratch the three architectures, i.e., MiniNet, Deeplabv3+ (a top-performing generic semantic segmentation approach) and ERFNet (a state-of-the-art for low-power GPUs that can also run on CPU) with the same configuration previously described.

Table 4.3 shows the performance of our approach and the other well-known architectures that we have trained on the same setup. The only difference is the image input resolution (which indeed affects directly the execution time), we show performance results with the resolution reported by the authors on their prior work: ERFNet 1024x512, Deeplabv3+ 512x512 and we set *MiniNet* to use 512x256. To enable more direct comparisons, note some of the relevant variations we have run and measured: ERFNet with 512x256 input takes 0.21 on CPU ($\times 8$ than *MiniNet*) and 0.008 on GPU ($\times 2$). ERFNet would need a 96x48 input to take the same time that *MiniNet* CPU forward pass.

Differently, from the previous experiment, the text detection quality metrics for *MiniNet* are much closer to the other approaches, and still present a huge CPU speed-up. This demonstrates that on this type of task *MiniNet* is able to get similar results

than state-of-the-art architectures for low-power GPUs while running x6-7 times faster on GPU and x30-60 times faster on CPU. Besides, the segmentation examples in Fig. 4.5, qualitatively confirm that MiniNet finds text regions similarly to state-of-the-art approaches. There are visual results with COCO-TEXT images as well as images from another public dataset: V4RL Urban Place Recognition Dataset [158], a challenging drone image dataset. It contains data from two drones mapping the environment and serves as a realistic robotic use case to evaluate the full system proposed in the next section. V4RL images were segmented with the CNNs trained on COCO-TEXT, without any adaptation on the model or the grayscale images.

Keyframe selection To evaluate the proposed keyframe selection we compare the relevance of the keyframes selected with it and with a state-of-the-art VSLAM algorithm, ORB-SLAM2. Typically VSLAM systems select keyframes online to perform the camera localization and store most of them to enable loop-detection/place-recognition tasks. We demonstrate how our keyframe selection method is fast enough to replace those selection strategies while it selects more relevant keyframes for additional high-level tasks to be performed on a robot team. We evaluate aspects of relevant information for place recognition, additional recognition tasks and quality of the selected images.

Set-up We use the V4RL Urban Place Recognition dataset [158], with outdoor data recorded for VSLAM and place recognition applications. The configuration parameters from Sec. 4.1.2 are set as follows. The candidate window size is set as half the distance between the last keyframe selected by our system and current keyframe selected by ORB-SLAM2. The window is placed in such a way that there are twice as many elements before the original keyframe than after. The number of selected top-Q frames, sorted according to the quality score, is 1/3 of the window size. These will then be processed on a batch through the segmentation CNN. With this configuration, the average cost per frame is just 7 ms on Intel i5-8600k and 16ms on the Intel NUC (i5-6260U). As an overview, the execution time (Intel Core i5-8600k) of each keyframe selection step for one image is: *Resize* 5ms, *Blurriness* score 0.1 ms, *Brightness* score 0.1 ms and *Semantic* score 18 ms.

Place recognition In these experiments, we evaluate the place recognition performance, i.e., the capability to relocate in an environment previously visited, since is an essential VSLAM capability that depends on the selected keyframes. We use the ground truth from V4RL dataset and evaluate the accuracy for place recognition. We use the DBoW2 [72] and match test query frames (from one of the drone sequences) to keyframes selected by ORB-SLAM2 algorithm or by our approach in the reference dataset (the other drone sequence, acquired at different day and time). Note we do not run a complete loop closure accuracy evaluation, but we focus on the semantic content of the keyframes. Therefore we provide accuracy of the top1 and top5 results provided by the DBoW2 algorithm. Fig. 4.6(b) shows that our proposal for selecting the keyframes does not lose any information with respect to the keyframes selected by ORB-SLAM2. Accuracy decreases equally for both approaches when the number of selected keyframes is reduced to less than 20% of the total amount of keyframes originally selected by the standard V-SLAM algorithm.

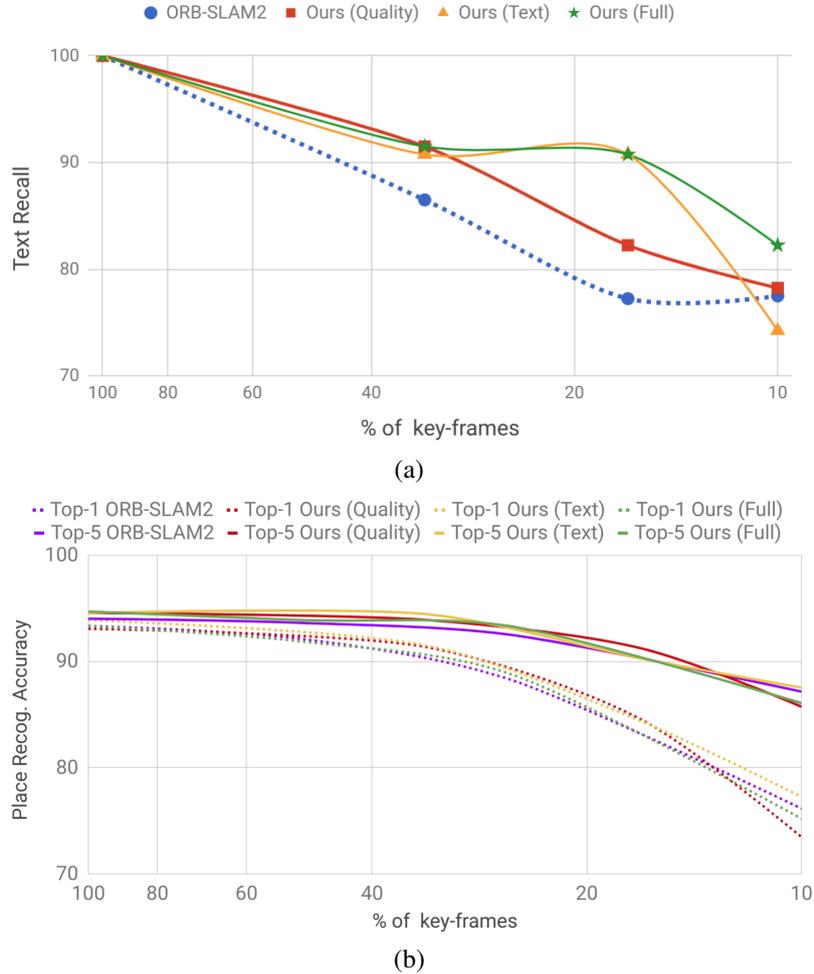


Figure 4.6: **Representativity of keyframe selection strategies.** (a) Average text (words) recognition recall with different % of keyframes. (b) Average place recognition accuracy (top1 and top5) obtained running DBoW2 algorithm with the different sets of selected keyframes.

Text Recognition This experiment shows that, besides maintaining performance in original tasks, our selected keyframes are more representative and useful for additional tasks. Since the V4RL dataset does not have any semantic label, we have manually labeled visible words in *Shopping Street 1 Sequence 1*, from four different intervals of 200 frames each: frames 50 to 250, 1250 to 1450, 3650 to 3850 and 6760 to 6960. Fig. 4.6 (a) shows the average recall of words correctly found running the text recognition from Gupta et al. [85] on keyframes selected by ORB-SLAM2 and by our approach. It shows separated results for each individual component of our score (*quality* and *semantic*) to verify the contribution of each to the final result (*Full*). The same text reading algorithm finds more information on our keyframes, regardless of the density of keyframes stored. This demonstrates our approach is more effective in selecting the data to share with the system. We would save computation and network resources by

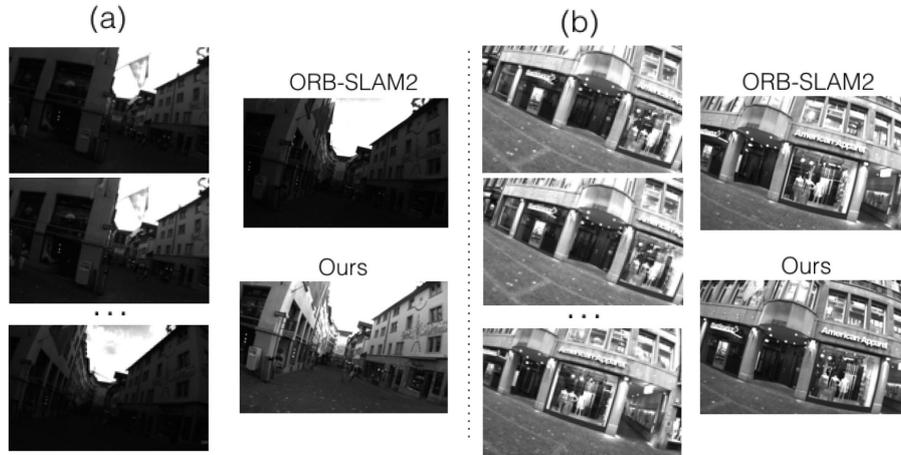


Figure 4.7: **Keyframes selected by our approach and ORB-SLAM2.** (a) Example with strong illumination changes, our approach picks keyframe with better contrast. (b) Example with large text-signs, our approach picks a keyframe with fully visible signs.

using the same keyframes to perform multiple tasks.

Qualitative results Our approach capabilities to enhance keyframe selection are highlighted in Fig. 4.7 (more examples available on the supplementary video). The first example shows how a better contrast frame is selected, which was just 7 frames far from the keyframe picked by ORB-SLAM2. Selecting this affects positively to the posterior image analysis. The second example shows the effect of our semantic score based on *MiniNet*: a frame containing two full shop signs is selected, as opposed to the keyframe picked by ORB-SLAM2, with one of them partially occluded.

4.1.5 Conclusion

We have presented a novel keyframe selection which can be integrated with state-of-the-art VSLAM systems to boost the usefulness of the keyframes. The benefits of our approach are particularly relevant for multi-robot systems or robots connected to a remote station since the proposed strategy allows the robots to be more efficient about what is shared with the team. The keyframes are selected considering multiple goals instead of purely based on the VSLAM criteria, which is what is commonly done in multi-robot mapping systems. Our experimentation with challenging drone imagery has shown that the proposed keyframe selection is more useful than the ORB-SLAM2 selection strategy. Evaluating the shared keyframes in the GPU-enabled server, we get better performance on additional tasks, text recognition in our experiments, while we do not lose the capacity of recognizing revisited places using those keyframes, essential for VSLAM systems.

A key ingredient in our approach is the efficient CNN proposed for image segmentation, which analyzes the frames online at the robot onboard CPU. This efficiency is essential to incorporate our selection algorithm without penalizing the other tasks run on the robot. Our experiments cover an in-depth analysis of the proposed CNN architecture, *MiniNet*. The good results with the presented *MiniNet* open opportunities for

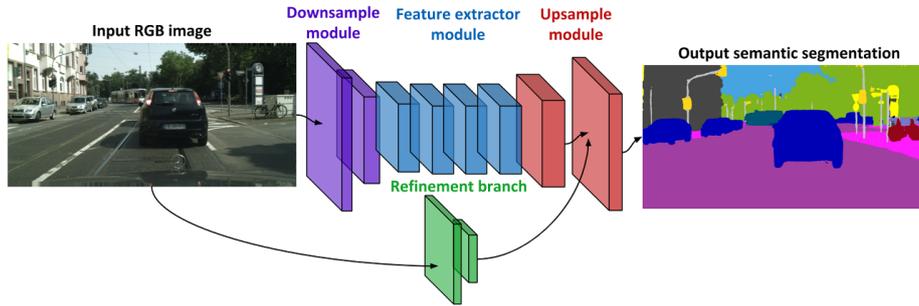


Figure 4.8: **MiniNet-v2: overview of the proposed architecture.**

additional applications based on quick video processing on low-resource nodes.

4.2 MiniNet: An Efficient Semantic Segmentation ConvNet for Real-time Robotic Applications

Research on novel efficient deep learning models is increasing the number of robotic applications that can make use of learning-based solutions. In particular, more efficient semantic segmentation models have brought a lot of attention [16, 28, 37, 283]. Semantic segmentation models compute semantic label probabilities for every image pixel, providing very rich information about the context and details within the captured scene. This task is essential for a quick scene understanding. As with many other deep learning-based solutions, it is unfeasible to run many of the top-performing models on CPUs, or even at low-powered GPUs, at the high rates required in many real applications.

CNN models usually require to have high-end GPUs to run inferences in near real-time. Unfortunately, these GPUs are often not available in robotic platforms or are not affordable for some applications running on small robots, drones or mobile phones. Even for autonomous vehicles, we cannot expect them to hold one high-end GPU for each task it has to perform. Therefore, there are many restrictions to take into account to deploy deep learning-based techniques in real-world applications. Available computational resources, power consumption and time restrictions are some of the significant constraints that robotic applications usually face. In the last years, we have seen an increasing interest in deep learning solutions for real-time applications on low-power GPUs. Several solutions [187, 202, 282] get results close to top-performing methods at much lower computational and energy cost.

This work presents a study of efficient techniques for efficient semantic segmentation that yielded to our novel architecture for efficient semantic segmentation. *MiniNet-v2* (summarized in Fig. 4.8 and later detailed in Table 4.4) presents a better accuracy vs resource requirements trade-off than the state-of-the-art. It is an improved version of *MiniNet*, firstly introduced in [10]. The main improvements come from the use of our proposed multi-dilation depthwise convolution and the use of an additional convolutional branch for retrieving fine-grained information instead of using skip connections. The new architecture presented here has two main configurations: *MiniNet-v2* is focused on more general set-ups with GPU availability and *MiniNet-v2-cpu* enables many applications and tasks to run semantic segmentation models without requiring

GPU on-board. For instance, we show its usefulness for V-SLAM smart key-frame selection, demonstrated in [10].

Our experiments evaluate how different CNN techniques and operations affect relevant semantic segmentation metrics, namely: GFLOPs (computation), the number of parameters and model size (memory), execution time (both on CPU and GPU) and Mean Intersection over Union (accuracy of the segmentation). We show that *MiniNet-v2* is more efficient overall than the state-of-the-art regarding those metrics, especially in the required memory, and still gets comparable performance on well known public datasets from autonomous driving environments, Cityscapes [50] and Camvid [29].

4.2.1 Related work

Techniques to improve CNNs efficiency Many works lately focus on reducing CNNs memory and computational requirements, which directly affects energy consumption and execution time. We next group some of the most relevant ideas into several categories depending on what they focus on.

Some approaches focus their contributions on the *training phase*. The most common strategy is joint training and distillation [93, 193]. These techniques rely on two models, one larger and not focused on efficiency which can get top-performing results, and a corresponding small and efficient model. The contribution relies on how to transfer the knowledge from the accurate model to the efficient one.

Other works focus their attention on carefully choosing the *parameter data types*. Quantized models [103] or binary networks [51] are recent works studying the effect of using less precise data types on the accuracy vs. memory trade-off and on the computation required.

Increased efficiency can also be achieved through *post-processing methods* such as pruning [174]. These methods study how to reduce a trained CNN without losing accuracy.

Other works study novel network *operations* such as depth-wise separable convolutions [217] and dilated convolutions [265]. Finally, other works propose novel CNN architectures directly focused on the target task. In particular, for efficient segmentation we find architectures such as Enet [187] or ERFNet [202]. These works propose different modules and strategies to perform semantic segmentation more efficiently.

Our work is more closely related to the last two groups of prior work: novel network operations and novel CNN architectures. Sec. 4.2.5 includes a more detailed analysis of existing network operations to improve efficiency on semantic segmentation architectures.

Semantic segmentation architectures CNNs for semantic segmentation typically follow an encoder-decoder structure: the encoder learns features while reducing the resolution and the decoder upsamples the learned features and maps them into the segmentation result. FCN [150] is one of the early works following a fully convolutional design for semantic segmentation. They propose to add a single decoder layer at the end of a classification CNN. Their results show that just upsampling the encoder features was enough to learn the semantic segmentation. SegNet [16] follows a symmetric encoder-decoder structure achieved by adding upsampling layers, i.e., unpooling. More recent works improve these earlier segmentation architectures by adding novel operations or modules proposed initially within CNNs architectures for classification

tasks. FC-DenseNet [112] follows the DenseNet work [101] using dense modules. PSPNet [283] uses ResNet [91] as its encoder and introduces the Pyramid Pooling Module, which is incorporated at the end of the CNN allowing to learn effective global contextual priors. Deeplab-v3 [37] is one of the top-performing architectures for segmentation and makes use of two powerful operations: the depthwise separable convolutions [217] and the atrous (or dilated) convolutions [265].

The semantic segmentation methods discussed so far have achieved impressive results but are not designed for computationally limited scenarios. Sometimes they rely on costly post-processing methods, e.g., CRFs or multi-scale inference, and they present too high computational requirements and inference time for embedded platforms.

Efficient segmentation architectures As efficient semantic segmentation architectures are concerned, ENet [187] sets up certain basis which following works, such as ERFNet [202], ICNet [282], ThunderNet [260] or GUN [162], have built upon. These architectures perform early downsampling of the feature maps so that most of the learning is performed at a low resolution (e.g., 1/8 of the input size). In these architectures, the computation is focused on the encoder which is in charge of learning the features and the decoder just upsamples them. In ERFNet [202], the authors propose an architecture inspired by Enet, but including the use of factorized convolutions [14], which reduce the number of learning parameters. ICNet [282] is a three-branch architecture that learns parameters at different resolutions and then joins the branches to compute the final result. GUN [162] is a two-branched architecture that also works at different resolutions but, differently from previous architectures, shares weights at early stages. It also propose a guided upampling operation, latter improved in Mazzini et al. [163]. Li et al. [146] perform an auto-search approach in stead of manual implemented modules to find an architecture with good speed-performance trade-off.

In the following section, we analyze and evaluate different techniques that can help building more efficient architectures.

4.2.2 Techniques for efficient semantic segmentation CNNs

This section presents a compilation of relevant ideas to build efficient semantic segmentation architectures and discusses their main insights.

Convolutional layers Convolutional layers are very relevant to our work since they are the computation core in CNNs. Apart from the standard convolutional layer, there are other layers that perform the convolution in a different way reducing the required number of parameters and operations. We consider the *factorized convolutions* and the *depthwise separable convolutions* [217] as the most relevant approximations for our goals. These ideas have been proved to perform very similar or even better than standard convolutions [202]. Figure 4.9 shows a comparison of the different types of convolutional layers.

Factorized convolution This convolutional layer consists of performing two consecutive convolutions factorizing the convolutional kernel. Let $W \in \mathbb{R}^{C_i \times d \times d \times C_o}$ denote the learning parameters of a 2D convolutional layer, where C_i is the number of input channels, C_o is the number of output channels and $d \times d$ represents the kernel size of

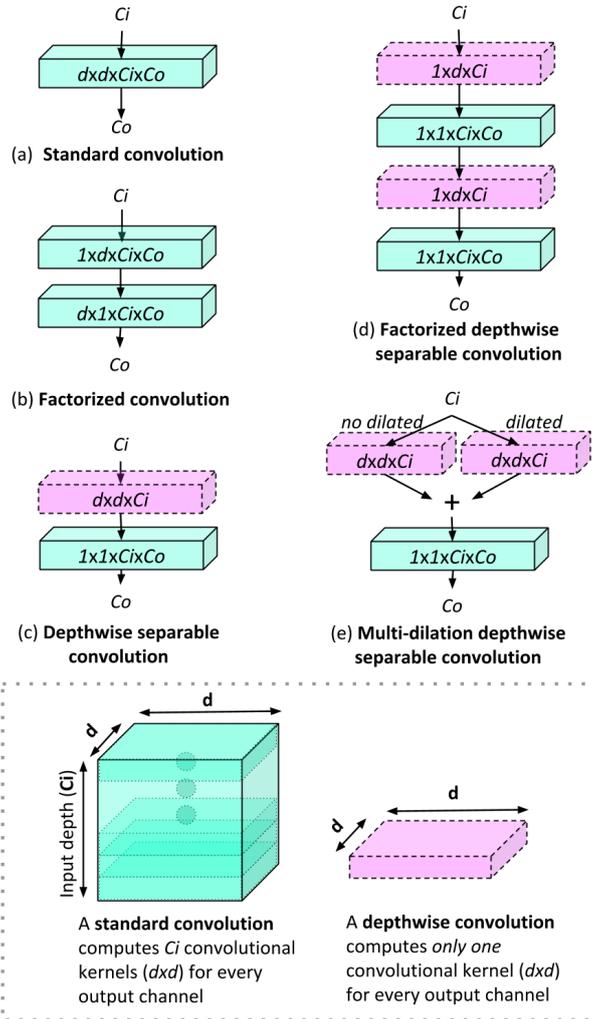


Figure 4.9: **Convolutions types.** Depiction of the different types of convolutions considered in this work, including required number of parameters on each step of the convolution (e.g., in $d \times d \times C_o \times C_i$: C_i is the number of input channels, C_o is the number of output channels and d is the kernel size).

each convolution. A standard 2D convolution has $d \times d \times C_i \times C_o$ learning parameters. In contrast, a *factorized convolution* layer has two convolutions of $1 \times d$ and $d \times 1$ kernels, leading to $2 \times d \times C_i \times C_o$ learning parameters. When setting a 3×3 kernel size, the parameter reduction comes to a 33% with respect to the standard convolutional layer.

Depthwise separable convolution It consists of factorizing the standard convolution into two separate convolutions. The first convolution is called the *depthwise convolution*. It performs a spatial convolution independently for each input channel, i.e., each output channel is only computed by one input channel in contrast to standard convolutions where all input channels are used for each output channel. Therefore, this *depth-*

wise convolution has $C_i \times d \times d$ parameters (where $C_i = C_o$). The second convolution is called the *pointwise convolution* which consists in performing a 1×1 convolution, projecting the output channels by the depthwise convolution onto a new channel space, combining the output of the *depthwise convolution*. It has $C_i \times C_o$ learning parameters. Thus, the total learning parameters is $C_i \times d \times d + C_i \times C_o$. Setting a 3×3 kernel size and 10^3 number of kernels leads to a reduction of 88% of the learning parameters with respect to the standard convolutional layer. Thus, the *depthwise separable convolution* reduces, even more, the number of parameters than the *factorized convolution*.

Combining both ideas, *depthwise separable convolution* and *factorized convolution*, yields to $(2 \times C_i \times d + C_i \times C_o)$ learning parameters, which leads to a reduction of 88% of learning parameters with respect to a standard convolutional layer.

Atrous convolution Atrous or dilated convolutions [265] introduce the dilatation rate r . This dilatation rate defines the stride between two adjacent kernel values. Therefore, a 3×3 kernel with $r = 2$ will have the same field-of-view as a 5×5 kernel, while only using 9 parameters instead of 25. When $r = 1$ there is no dilation and it is just a standard convolution.

This type of convolution is very important regarding efficient semantic segmentation because it allows lowering the number of layers and parameters [187, 202, 282] while being able to cover the same field-of-view. Nevertheless, this type of convolutions are not fully established in semantic segmentation architectures because big dilation rates lose local information.

Multi-dilation depthwise separable convolution We introduce this new convolutional layer, which consists of two parallel depthwise convolutions, one with a dilatation rate $r = 1$ and the other one with $r \geq 1$. Then, their outputs are added and then a pointwise convolution is applied as Fig. 4.9 shows.

This convolutional layer improves the performance by efficiently learning both local and global spatial relationships. It learns a larger variety of kernels with less number of layers thanks to the dilatation rate. This layer only adds $C_i \times d \times d$ parameters with respect to the standard depthwise separable convolution. Therefore, it still leads to a reduction of 87% of the learning parameters with respect to the standard convolutional layer.

Other efficient techniques to explore There are additional techniques that can improve efficiency in semantic segmentation architectures, which we discuss next.

Retrieving fine-grained information Encoder-decoder architectures for CNNs reduce the resolution of the input image (between $\times 8$ and $\times 32$) when learning the features in the encoder. Working at low resolutions hinders the CNN to get detailed outputs because it does not learn well fine-grained information. Another fact that makes difficult to learn fine-grained information is stacking too many convolutional layers, because deep features encode the image context rather than local and spatial information. There are two common strategies to deal with this issue.

Several works [112, 205] connect early layers to final layers, in order to keep high-resolution information and recover fine-grained information without the need of adding more layers to the network. Nevertheless, these skip connections have some drawbacks: early layers have to extract fine-grained information for enhancing the output

and useful low-level features for the encoder, which, made at the same time, may hinder the optimization.

Another strategy towards the same goal is to have a very light independent branch of convolutions for spatial information preservation [269]. This second strategy does not force early layers to extract different types of features, but adds more computational cost. Section 4.2.5 includes a quantitative comparison of these strategies.

Reducing output resolution Working on high resolutions implies a high computational cost. Therefore, apart from working at low resolutions at the encoder, another way to save computational resources is to perform the semantic segmentation predictions on a lower resolution than the input resolution removing last convolution layers and upsampling layers. This strategy has been shown to provide significant computational saving with little loss in accuracy [37].

4.2.3 Proposed Efficient semantic segmentation architecture

This section describes our architecture for efficient semantic segmentation: *MiniNet-v2*³. It is inspired by ERFNet [202] and our design choices are based on the benefits and drawbacks of the techniques discussed in previous section.

Main blocks The main blocks are shown in Table 4.4, where the whole architecture is defined. The key ingredient and the core of MiniNet-v2 architecture is its *convolutional module*. It consists of a 3×3 depthwise separable convolution followed by a residual connection. If the module is applied on the Feature Extractor block, we use multi-dilation depthwise separable convolution instead and add a dropout of 0.25 after the convolution. All *downsample* operations consist of a max-pool operation concatenated with a strided convolution and all *upsample* operations are transposed convolutions.

Downsample block This block quickly downsamples the features. It consists of combining downsample operations with convolutional modules.

Feature extractor block This block is the main part of the encoder. It consists of several consecutive convolutional modules with different dilation rates.

Refinement block This block extracts spatial and high resolution features performing two downsample operations to the input image. This block goal is to extract additional features that can help to refine the features previously learned in the feature extractor block.

Upsample block This block upsamples the feature extractor block output. Then, the refinement block and this block are added. The last part of this block consists of several convolutional modules without dilation rate and upsample operations. The output resolution is half of the input's resolution.

³Code at <https://github.com/Shathe/MiniNet-v2>

MiniNet-v2 This new architecture enhances our previous one, firstly introduced in [10], and gets similar performance than state-of-the-art models with much larger number of parameters and memory requirements. We propose two different configurations: the main architecture, MiniNet-v2, defined in Table 4.4, and a smaller version, MiniNet-v2-cpu, built for CPU applications. This smaller version is the same architecture removing these convolutional modules: from m3 to m10, from m16 to m25 and from m28 to m29.

MiniNet-v2 has 0.52M parameters and the model takes up to 2.02MB. At 1024x512 resolution, it works at 50fps in a TitanXp and 4fps in an Intel i5-8600k, and performs 12.89 GFLOPs (floating point operations) and 6.45 GMACs (multiply-accumulate operations), getting 70.4% MIOU on the Cityscapes benchmark.

MiniNet-v2-cpu has 0.27M parameters and the model takes up to 1.06MB. At 512x256 resolution, it works at 250fps in a TitanXp and 30fps in an Intel i5-8600k, and performs 1.68 GFLOPs and 0.84 GMACs, getting 59.9% MIOU on the Cityscapes benchmark. This smaller version gets lower segmentation accuracy but is more efficient due to smaller input resolution and the reduction on the number of convolutional layers.

4.2.4 Evaluation

We use the following metrics and datasets for the evaluation.

Metrics

- **Execution time.** Inference time both on GPU (on Titan Xp using PyTorch framework) and CPU (on Intel i5-8600k).
- **Memory.** Number of parameters of the CNN and the required memory for the model (MB).
- **Computation.** GFLOPs (Giga Floating Point Operations) of a forward step.
- **MIOU** (Mean Intersection over Union). This is the most common metric for semantic segmentation tasks.

Datasets

- **Cityscapes** [50]. An urban-scene understanding dataset widely adopted to evaluate semantic segmentation approaches. It consists of 19998 coarsely annotated images and 5000 fine-annotated images (split into 2975 images for training, 500 images for validation, and 1525 images for testing). Test set evaluation is performed by submitting the test predictions on the official benchmark server.

- **Camvid** [29]. An autonomous driving dataset frequently used to train existing state-of-the-art approaches for urban areas image segmentation. It consists of 367 training images, 101 validation images, and 233 test images.

- **COCO-Text** [243]. A text detection dataset based on the MS COCO dataset. It contains over 173k text annotations in over 63k images. The dataset provides bounding boxes for the text detection task, but we convert them into pixel labels, i.e., binary segmentation (text vs non-text).

4.2.5 Analysis of MiniNet-v2 components

This section evaluates the effects of the described techniques in Sec. 4.2.2 and justifies our design of *MiniNet-v2*.

Table 4.4: *MiniNet-v2* Architecture for an input size of 1024x512.

	Block	Name	Type	Input	Output size	
Downsample		d1	downsampling	image	512x256x16	
		d2	downsampling	d1	256x128x64	
		m1	rate=1	d2	256x128x64	
		m2	rate=1	m1	256x128x64	
		m3	rate=1	m2	256x128x64	
		m4	rate=1	m3	256x128x64	
		m5	rate=1	m4	256x128x64	
		m6	rate=1	m5	256x128x64	
		m7	rate=1	m6	256x128x64	
		m8	rate=1	m7	256x128x64	
		m9	rate=1	m8	256x128x64	
Feature extractor		m10	rate=1	m9	256x128x64	
		d3	downsampling	m10	128x64x128	
		m10	rate=1	d3	128x64x128	
		m11	rate=2	m10	128x64x128	
		m12	rate=1	m11	128x64x128	
		m13	rate=4	m12	128x64x128	
		m14	rate=1	m13	128x64x128	
		m15	rate=8	m14	128x64x128	
		m16	rate=1	m15	128x64x128	
		m17	rate=16	m16	128x64x128	
		m18	rate=1	m17	128x64x128	
		m19	rate=1	m18	128x64x128	
		m20	rate=1	m19	128x64x128	
		m21	rate=2	m20	128x64x128	
		m22	rate=1	m21	128x64x128	
		m23	rate=4	m22	128x64x128	
		m24	rate=1	m23	128x64x128	
		m25	rate=8	m24	128x64x128	
	Ref		d4	downsampling	image	512x256x16
			d5	downsampling	d4	256x128x64
	Upsample		up1	upsampling	m25	256x128x64
			m26	rate=1	up1 + d5	256x128x64
			m27	rate=1	m26	256x128x64
			m28	rate=1	m27	256x128x64
			m29	rate=1	m28	256x128x64
		output	upsampling	m29	512x256xN	

rate: stands for the convolutional dilation rate.

Training protocol and implementation details The training protocol is the same for all our experiments. We train the different CNN configurations on the Cityscapes data (only fine-annotated images) for 250 epochs with a batch size of 12. We use Adam optimizer with an initial learning rate of 10^{-3} and polynomial learning rate decay schedule which power is set to 0.9. We use a weight decay of 2×10^{-4} . We use horizontal flips and shifts for data augmentation. As similar architectures, we perform the training optimization via back-propagation of the loss. The loss is calculated as the

Table 4.5: Performance of factorized (ERFNet) and non-factorized (MiniNet-v2) convolutions using depthwise separable convolutions (D) or standard convolutions (S).

Configuration	GPU (ms)	CPU (ms)	Params (M)	GFLOPs	Memory (MB)	MIoU
ERFNet-S [202]	10	88	2.06	13.42	7.95	58.79
ERFNet-D	10	62	0.49	3.20	1.93	58.36
MiniNet-v2-S	9	109	3.02	19.68	11.55	58.89
MiniNet-v2-D	9	61	0.49	3.28	1.97	58.51

sum of all per-pixel losses, through parameter gradients using the common soft-max cross entropy loss function.

For these experiments, MiniNet-v2 components are evaluated using the Cityscapes validation set with input resolution of 512x256 (half of its original input size).

Techniques for efficient semantic segmentation CNNs In the following experiments, we refer as *MiniNet-v2* (basic version) to our initial ERFNet modification consisting of replacing each ERFNet convolutional module with two *MiniNet-v2* convolutional modules (explained in Sec.4.2.3).

Factorized convolutions and depthwise separable convolutions This is the most important experiment because it is focused on variations on convolution layers, which are the main ingredient in CNNs. This experiment compares standard convolutions (S), depthwise separable convolutions (D) and multi-dilation separable convolutions (M). It also compares factorized convolutions used by ERFNet with respect to non-factorized convolutions used by *MiniNet-v2*.

Table 4.5 summarizes the results of this evaluation. The first thing to note is that standard convolutions (S) have too many parameters, FLOPs and execution time compared to depthwise convolutions. Regarding factorized convolutions, they take more execution time and present worse segmentation performance (MIoU) than non-factorized convolutions, while having the advantage of running slightly fewer FLOPs.

The approach we select for our architecture (and used in the following experiments), MiniNet-v2-D, shows several enhancements over the original ERFNet architecture (ERFNet-S): it reduces the generated computation (GFLOPs) by a 75% and the memory cost (number of parameters) by a 75%.

Retrieving fine-grained information This experiment compares two different strategies to preserve high-resolution information and recover fine-grained information from the input image.

We consider two different configurations. In the first setup (S), we sum the output of the first downsampling operation and the output of the penultimate upsampling operation. The second configuration, a new convolutional branch (I), performs two downsample operations to the input image and then adds it to the penultimate upsampling operation.

Table 4.6 confirms our hypothesis on the detrimental effect of the first skip connection configuration (S). This effect is caused by forcing early layers to perform two jobs at the same time: early layers have to extract fine-grained information for enhancing the output and useful features for the rest of the encoder. This is especially important

Table 4.6: Performance comparison between skip connection from early layers (S) versus a light additional branch (I).

Configuration	GPU (ms)	CPU (ms)	Params (M)	GFLOPs	Memory (MB)	MIoU
MiniNet-v2-D	9	61	0.49	3.28	1.97	58.51
MiniNet-v2-D-S	9	62	0.49	3.28	1.97	56.89
MiniNet-v2-D-I	9	62	0.49	3.42	2.00	58.71

Table 4.7: Performance when varying dilation rates: ERFNet default dilation (E), doubling ERFNet dilation (2), No dilation (0), minimum dilation for whole field-of-view (1) or using Multi-dilation depthwise convolutions (M).

Configuration	GPU (ms)	CPU (ms)	Params (M)	GFLOPs	Memory (MB)	MIoU
MiniNet-v2-D-I-E	9	61	0.49	3.42	2.00	58.71
MiniNet-v2-D-I-0	9	55	0.49	3.42	2.00	57.91
MiniNet-v2-D-I-1	9	60	0.49	3.42	2.00	58.81
MiniNet-v2-D-I-2	9	64	0.49	3.42	2.00	57.21
MiniNet-v2-D-I-M	10	69	0.53	3.66	2.10	59.36

in small and efficient networks where there are very few layers and fewer parameters. In contrast, the additional convolutional branch (I) allows the network to extract fine-grained information without any negative effect. This information allows the network to get more accurate outputs, especially in contours, which is beneficial to classes with small size like traffic lights.

Atrous convolutions Atrous or dilated convolutions allow adjusting the kernel field-of-view to capture multi-scale information. Nevertheless, they increase the execution time. In this experiment, we evaluate four different configurations: same dilation rates as ERFNet (E); double dilation rates than ERFNet (2); no dilated convolutions, (0); using minimum dilation rates needed to reach a field-of-view equal to the feature extractor input resolution (1). We also evaluate our proposed convolutional layer, multi-dilation depthwise separable convolution, which performs two depthwise convolutions in parallel: one with dilation rate and another one without it.

As Table 4.7 shows, decreasing the dilation rates harms the MIoU performance while it does not improve much the efficiency. This is particularly noticeable when there are not many convolutional layers, as it usually happens in efficient architectures. This is due to a lack of context information. Increasing too much the dilation rates has the opposite effect, it lacks local information. Our architecture uses multi-dilation separable convolutions, which outperform the other methods. The improvement is mostly due to the effect of learning global and local context at the same time, thanks to the two parallel depthwise convolutions with different dilation rates.

Reducing output resolution This experiment evaluates the benefits of reducing the output resolution with respect to the input’s resolution. We compare the MiniNet-

Table 4.8: Performance when reducing the output resolution (R).

Configuration	GPU (ms)	CPU (ms)	Params (M)	GFLOPs	Memory (MB)	MIoU
MiniNet-v2-D-I-M	10	74	0.53	3.66	2.10	59.36
MiniNet-v2-D-I-M-R	8	54	0.52	3.22	2.02	59.08

v2-D-I-M configuration (the best so far) with MiniNet-v2-D-I-M-R, which removes the penultimate upsampling operation and the four last *MiniNet-v2* convolutional modules that were performed at that last resolution. We only decreased the resolution $\times 2$ to avoid losing too much information.

Table 4.8 shows that the execution time has been drastically reduced, 33% for GPU and 22% for CPU, because higher resolutions take most of the computation. Besides, the drop in the MIoU performance is not very significant. For these reasons, our proposed *MiniNet-v2* architecture follows this approach. This last configuration, MiniNet-v2-D-I-M-R, is the *MiniNet-v2* final architecture.

4.2.6 Benchmark evaluation

This section compares our work to current state-of-the-art on different semantic segmentation problems, using Cityscapes, Camvid, and COCO-Text datasets.

Multi-class segmentation

Training protocol For the Cityscapes dataset, differently from the previous section, we jointly train on the coarse-annotated and fine-annotated data. For all datasets, we train for 1M iterations with a batch size of 6, an initial learning rate of 1×10^{-3} , polynomial learning rate decay schedule which power is set to 0.9 and weight decay of 2×10^{-4} . We use horizontal flips, random scaling ($\times 0.5$, $\times 2$) and horizontal shifts for data augmentation. For the optimization we use the cross entropy loss function. To account for class imbalance, we use the median frequency class balancing, as applied in SegNet [16]. To smooth the resulting class weights, we propose to apply a power operation: $w_c = (\frac{f_{median}}{f_c})^i$, with f_c being the frequency of class c and f_{median} the median of all frequencies. We set i to 0.12. When computing the loss, instead of resizing the labels to match the output shape of our network, we resized the output for not losing information in the labeled image.

Evaluation Table 4.9 shows the results on the test set of the Cityscapes dataset and Figure 4.10 shows visual results. *MiniNet-v2* at 1024x512 resolution uses $\times 4$ less memory, parameters and FLOPs than ERFNet (our main baseline) while getting higher performance (MIoU). Our architecture gets state-of-the-art performance (MIoU) for efficient semantic segmentation, similar to ICNet or Li et al. with pretraining, but with the lowest memory requirements reported. At this resolution, it also gets better performance than other networks that require many more parameters like GUN, ThunderNet or SegNet. When it comes to *MiniNet-v2* at 512x256 resolution, the MIoU performance is higher (+5 IoU) than ESPNet and ESPNet using similar number of parameters, memory, FLOPs and runtime. The MIoU reached is also higher than ThunderNet's

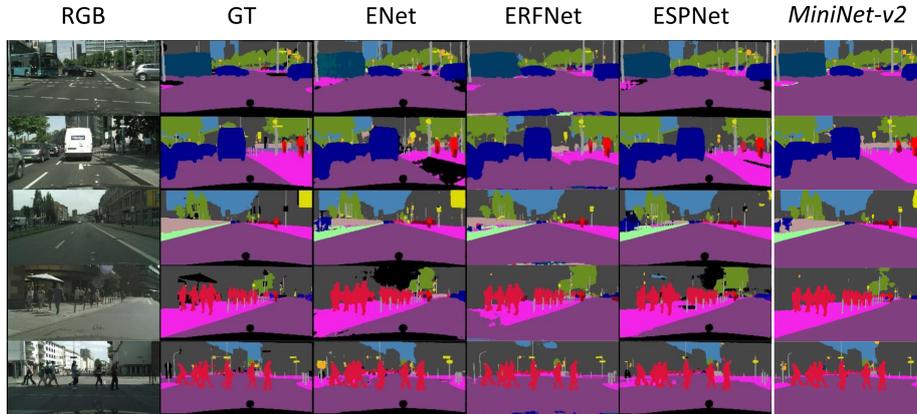


Figure 4.10: **Qualitative results on the Cityscapes validation dataset.** We show that *MiniNet-v2* achieves really accurate segmentation results, even better than state-of-the-art methods. Different colors mean different classes (except black, which is the ignore/void class). Visual results taken from [165] except ours.

(pre-trained) and Mazzini et al. (without pretraining) MIoU, with similar efficiency metrics. Our *MiniNet-v2-cpu* architecture, which is focused on CPU applications, gets similar MIoU than ENet and ESPNet while presenting better efficiency metrics.

Table 4.10 shows a comparison of our models with the state-of-the-art using the Camvid benchmark. *MiniNet-v2* at 960x720 resolution presents a little higher MIoU than ICNet and ERFNet, and it is much more efficient in terms of both memory and speed. Comparing methods at 480x360 resolution, *MiniNet-v2* gets higher MIoU than most of the methods (Enet, SegNet, FC-DenseNet56) and similar results than other

Table 4.9: Evaluation on the **test set** of the **Cityscapes** dataset.

Method	GPU ms (type)	Input Resolution	Params (M)	Memory (MB)	GFLOPs	MIoU w pt	MIoU w/o pt
Li et al. [141]	7* (1080ti)	1536x768	2.9**	—	—	71.4	—
ICNet [282]	30 (TX)	2048x1024	6.7	—	—	—	70.6
GUN [162]	27 (TXp)	1024x512	19.0	—	58.7	70.4	—
ERFNet [202]	20 (TXp)	1024x512	2.1	7.95	53.78	69.7	68.0
Mazzini et al. [163]	9 (TXp)	1024x512	1.4	—	—	68.9	63.7
CGNet [258]	56 (V100)	2048x1024	0.5	—	—	—	64.8
ThunderNet [260]	10 (TX)	1024x512	4.7	—	—	64.0	—
ESPNet [165]	9 (TX)	512x256	0.4	1.46	4.5	—	60.3
ENet [187]	13 (TX)	1024x512	0.4	1.64	8.72	—	58.3
SegNet-basic [16]	60 (TX)	480x360	29.5	112.40	286.03	—	56.1
MiniNet [10]	5 (TXp)	512x256	3.1	12.35	1.06	—	40.7
<i>MiniNet-v2</i>	20 (TXp), 11 (2080ti)	1024x512	0.5	2.02	12.89	—	70.5
<i>MiniNet-v2</i>	9 (TXp)	512x256	0.5	2.02	3.22	—	64.7
<i>MiniNet-v2-cpu</i>	4 (TXp)	512x256	0.3	1.06	1.68	—	59.3

w pt: with pretraining on Imagenet; TX: TitanX; TXP: TitanXP

* Reported using TensorRT. 25 ms using Caffe Time.

** Only encoder measure is reported.

Table 4.10: Evaluation on the **test set** of the **Camvid** dataset.

Method	GPU ms (type)	Input Resolution	Params (M)	Memory (MB)	GFLOPs	MIoU w pt	MIoU w/o pt
Mazzini et al. [163]	4(TXp)	480x360	1.4	—	—	68.7	—
ERFNet [203]	28 (TXp)	960x720	2.1	7.95	70.90	—	68.3
ICNet [282]	36 (TX)	960x720	6.7	—	—	—	67.1
FC-DenseNet103 [112]	109 (TXp)	480x360	9.4	35.77	139.43	—	66.9
FC-DenseNet56 [112]	70 (TXp)	480x360	1.5	5.29	61.75	—	58.9
SegNet-basic [16]	60 (TX)	480x360	29.5	112.4	286.03	55.6	—
ENet [187]	6 (TX)	480x360	0.4	1.64	2.87	—	51.3
MiniNet [10]	5 (TXp)	512x256	3.1	12.35	1.06	—	41.3
<i>MiniNet-v2</i>	28 (TXp), 15 (2080ti)	960x720	0.5	2.02	17.01	—	69.0
<i>MiniNet-v2</i>	9 (TXp)	480x360	0.5	2.02	4.22	—	66.1
<i>MiniNet-v2-cpu</i>	5 (TXp)	480x360	0.3	1.06	2.22	—	59.9

w pt: with pretraining on Imagenet. TX: TitanX; TXP: TitanXP

Table 4.11: **Binary segmentation** results on the COCO-Text dataset.

Method	GPU ms (type)	Input Resolution	Params (M)	Memory (MB)	GFLOPs	MIoU
DeeplabV3+ [37]	32 (TXp)	512x512	41.1	—	51.44	32.29
ERFNet [202]	20 (TXp)	1024x512	2.1	7.95	53.78	29.27
MiniNet [10]	5 (TXp)	512x256	3.1	12.35	1.06	27.63
<i>MiniNet-v2</i>	20 (TXp), 11 (2080ti)	1024x512	0.5	2.02	12.89	30.78
<i>MiniNet-v2</i>	8 (TXp)	512x256	0.5	2.02	3.22	29.02
<i>MiniNet-v2-cpu</i>	4 (TXp)	512x256	0.3	1.06	1.68	28.82

There is no pretraining run for any of the methods

methods that use many more parameters, like FC-DenseNet103. Considering only the fastest approaches, *MiniNet-v2-cpu* gets better performance (+8.6% MioU) than ENet and earlier MiniNet (+18.6% MioU) and presents much lower memory requirements. It also beats some relevant architectures, like FC-DenseNet56, in all the metrics. *MiniNet-v2-cpu* gets similar performance than ENet in the Cityscapes dataset, but, interestingly, it gets significantly better performance in the Camvid data. This could mean that our architecture learns better on smaller datasets, a common case in robotics. Note that this dataset is $\times 8$ smaller than the Cityscapes dataset and therefore, it is more sensitive to pretraining: MiniNet-v2 goes from 69% MioU to 76% in the Camvid data if pre-trained on Cityscapes.

Binary segmentation Previous experiments have shown how *MiniNet-v2* gets similar or better results than state-of-the-art models while being more efficient regarding several metrics. However, *MiniNet-v2-cpu* is 10 MioU points below when evaluated in the multi-class benchmarks from the two previous experiments. Differently from previous experiments performed on datasets with more semantic classes, now we consider binary text segmentation. We show that for simpler semantic segmentation tasks, *MiniNet-v2-cpu* also gets similar results than top-performing CNNs, dropping only 2 MioU points from *MiniNet-v2*.

For this experiment, we use the COCO-Text dataset [243] (a subset of machine printed and legible text images). We trained four architectures: *MiniNet-v2-cpu*, MiniNet-

v2, Deeplabv3+ (top-performing generic semantic segmentation approach) and ERFNet (the state-of-the-art for low-power GPUs), using the same training protocol as in previous experiments.

Table 4.11 shows that our approach reaches comparable performance to state-of-the-art architectures (even including those not focused on efficiency like Deeplab) when training for simpler segmentation tasks. Even *MiniNet-v2-cpu* performs similar to top methods, while running at 250fps on GPU and 30fps on CPU.

4.2.7 Conclusions

This work presents our novel architecture for efficient semantic segmentation, *MiniNet-v2*, and its variation *MiniNet-v2-cpu*, improving our earlier work in *MiniNet*. These models pave the way for applications in robotics or related embedded systems that require quick visual scene understanding steps. Our model design is based on the conclusions from the presented study about the most relevant techniques to build efficient CNNs. We analyze the trade-offs provided by several techniques and operations for efficient semantic segmentation. Our results show that our proposed multi-dilation depthwise separable convolutions, the use of an additional convolutional branch instead of skip connections and reducing the output resolution, are key steps to achieve a good trade-off between accuracy and computational requirements. Our proposed architectures have been thoroughly evaluated to demonstrate their benefits. *MiniNet-v2* achieves similar or better results than state-of-the-art models on known public benchmarks for semantic segmentation, while it provides lower memory and computation requirements. *MiniNet-v2-cpu* can be used for real-time CPU-only applications, achieving similar results than top-performing CNN architectures when the segmentation task is not very complex, e.g., binary segmentation. Besides, in previous Section 4.1, we have shown a proof of concept demonstration of the benefits of these models in robotic applications, namely, to allow a robot to quickly analyze the content of a video to decide which frames to share with the rest of the robotic team. The availability of all our code leaves the door open for the development of additional real-time applications using our models.

Chapter 5

Semantic Segmentation With Other Sensors

So far, all the proposed solutions for semantic segmentation challenges in previous chapters have been designed and evaluated for RGB camera data. However, in many scenarios, for example in robotics, other sensors are available and can provide additional useful information. This last part of the thesis focuses on how to extract semantic information from data acquired from two very relevant sensors in the last years: LiDAR and event-based cameras.

The first part of the chapter tackles the semantic segmentation of LiDAR point clouds. 3D data, like LiDAR point clouds, is a more complex data structure than RGB images, since they are not as homogeneously distributed and well-structured as image pixels. The first contribution in this topic is the extension of the previously proposed efficient 2D semantic segmentation models to LiDAR point clouds. The second contribution copes with the domain adaptation problem for LiDAR semantic segmentation, i.e., try to minimize the drop of performance when model trained on certain domain is applied to data from a different domain.

The last section of this chapter addresses the semantic segmentation problem using event-based camera data. This is a novel sensor that provides some advantages over RGB cameras but also brings additional challenges. For example, it is particularly difficult to label this type of data for humans.

5.1 3D-MiniNet: Fast and Efficient 3D LiDAR Semantic Segmentation

Autonomous robotic systems use sensors to perceive the world around them. RGB cameras and LIDAR are very common due to the essential data they provide. One of the key building blocks of autonomous robots is semantic segmentation. Semantic segmentation assigns a class label to each LIDAR point or camera pixel. This detailed semantic information is essential for decision making in real-world dynamic scenarios. LIDAR semantic segmentation provides very useful information to autonomous robots when performing tasks such as Simultaneous Localization And Mapping (SLAM) [113, 284], autonomous driving [169] or inventory tasks [41], especially for identifying dynamic objects. In these scenarios, it is critical to have models that pro-

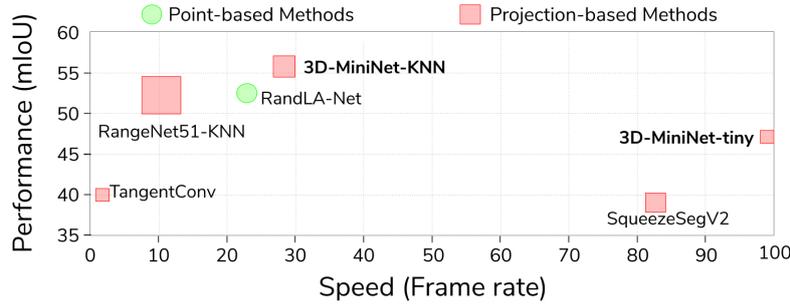


Figure 5.1: 3D LIDAR semantic segmentation **accuracy vs speed** on SemanticKITTI test set [19]. **Green** circles depict point-based methods and **red** squares are projection-based methods. Area of these shapes is proportional to the method number of parameters. The proposed 3D-MiniNet outperforms previous methods with less parameters and faster execution.

vide accurate semantic information in a fast and efficient manner, which is particularly challenging working with 3D LIDAR data. On one hand, the commonly called *point-based approaches* [190,191,232] tackle this problem directly executing 3D point-based operations, which is computationally expensive to operate at high frame rates. On the other hand, approaches that project the 3D information into a 2D image (*projection-based approaches*) are more efficient [58, 169, 251, 256, 257] but do not exploit the raw 3D information. Recent results on fast [169] and parameter-efficient [281] semantic segmentation models are facilitating the adoption of semantic segmentation in real-world robotic applications [19, 140].

This work presents a novel fast and parameter-efficient approach for 3D LIDAR semantic segmentation that consists of three modules (as detailed in Sec. 5.1.2). The main contribution relies on our 3D-MiniNet module. 3D-MiniNet runs the following two steps: (1) It learns a 2D representation from the 3D point cloud (following previous works on 3D object detection [134,286,287]); (2) It computes the segmentation through a fast 2D fully convolutional neural network.

Our best configuration achieves state-of-the-art results in well known public benchmarks (SemanticKITTI [19] and KITTI dataset [75]) while being faster and more parameter efficient than prior work. Figure 5.1 shows how 3D-MiniNet achieves better precision-speed trade-off than previous methods. The main novelties with respect to existing approaches, that facilitate these improvements, are:

- An extension of MiniNet-v2 for 3D LIDAR semantic segmentation: 3D-MiniNet.
- Our novel projection module.
- A validation of 3D-MiniNet on the SemanticKITTI benchmark [19] and KITTI dataset [75].

The proposed projection module learns a rich 2D representation through different operations. It consists of four submodules: a context feature extractor, a local feature extractor, a spatial feature extractor and the feature fusion. We provide a detailed ablation study on this module showing how each proposed component contributes to improve the final performance of 3D-MiniNet. Besides, we implemented a fast version of the point neighbor search based on a sliding-window on the spherical projection [25]

in order to compute it at an acceptable frame-rate. All the code and trained models are available online ¹.

5.1.1 Related Work

5.1.1.1 2D Semantic Segmentation

Current 2D semantic segmentation state-of-the-art methods are deep learning solutions [35, 36, 112, 150]. Semantic segmentation architectures are evolved from convolutional neural networks (CNNs) architectures for classification tasks, adding a decoder on top of the CNN. Fully Convolutional Neural Networks for Semantic Segmentation (FCNN) [150] carved the path for modern semantic segmentation architectures. The authors of this work propose to upsample the learned features of classification CNNs using bilinear interpolation up to the input resolution and compute the cross-entropy loss per pixel. Another of the early approaches, SegNet [16], proposes a symmetric encoder-decoder structure using the unpooling operation as upsampling layer. More recent works improve these earlier segmentation architectures by adding novel operations or modules proposed initially within CNNs architectures for classification tasks. FC-DenseNet [112] follows DenseNet work [101] using dense modules. PSPNet [283] uses ResNet [91] as its encoder and introduces the Pyramid Pooling Module incorporated at the end of the CNN allowing to learn effective global contextual priors. Deeplab-v3+ [36] is one of the top-performing architectures for segmentation. Its encoder is based on Xception [45], which makes use of depthwise separable convolutions [217] and atrous (dilated) convolutions [265].

With respect to efficiency, ENet [187] set up certain basis which following works, such as ERFNet [202], ICNet [282], have built upon. The main idea is to work at low resolutions, i.e., quick downsampling, and to focus the computation on the encoder having a very light decoder. MiniNet-v2 [11] uses a multi-dilation depthwise separable convolution, which efficiently learns both local and global spatial relationships. In this work, we take MiniNet-v2 as our backbone and adapt it to capture information from raw LIDAR points.

5.1.1.2 3D Semantic Segmentation

There are three main groups of strategies to approach this problem: point-based methods, 3D representations and projection-based methods.

Point-based Methods Point-based methods work directly on raw point clouds. The order-less structure of the point clouds prevents standard CNNs to work on this data. The pioneer approach and base of the following point-based works is PointNet [190]. PointNet proposes to learn per-point features through shared MLP (multi-layer perceptron) followed by symmetrical pooling functions to be able to work on unordered data. Lots of works have been later proposed based on PointNet. Following with the point-wise MLP idea, PoinNet++ [191] groups points in an hierarchical manner and learns from larger local regions. The authors also propose a multi-scale grouping for coping with the non-uniformity nature of the data. In contrast, other approaches propose different types of operations following the convolution idea. Hua et al. [100] propose to bin neighboring points into kernel cells for being able to perform point-wise convolutions. Other works resort to graph networks to capture the underlying geometric

¹<https://sites.google.com/a/unizar.es/semanticseg/>

structure of the point cloud. Loic et al. [133] use a directed graph to capture the structure and context information. For this, the authors represent the point cloud as a set of interconnected superpoints.

3D representations There are different kinds of representations of the raw point cloud data which have been used for 3D semantic segmentation. SegCloud [233] makes use of a *volumetric or voxel representation*, which is a very common way for encoding and discretizing the 3D space. This approach feeds the 3D voxels into a 3D-FCNN [150]. Then, the authors introduce a deterministic trilinear interpolation to map the coarse voxel predictions back to the original point cloud and apply a CRF as a final step. The main drawback of this voxel representation is that 3D-FCNN has very slow execution times for real-time applications. Su et al. [224] proposed SPLATNet, making use of another type of representation: *Permutohedral Lattice representation*. This approach interpolates the 3D point cloud to a permutohedral sparse lattice and then bilateral convolutional layers are applied to convolve on occupied parts of the representation. LatticeNet [207] was later proposed improving SPLATNet proposing its DeformSlice module for re-projecting the lattice feature back to the point cloud.

Projection-based Methods This type of approaches rely on projections of the 3D data into a 2D space. For example, TangentConv [232] proposes to project the neighboring points into a common tangent plane where they perform convolutions. Another type of projection-based method is the *spherical representation*. This strategy consists of projecting the 3D points into a spherical projection and has been widely used for LIDAR semantic segmentation. This representation is a 2D projection that allows the application of 2D images operations, which are very fast and work very well on recognition tasks. SqueezeSeg [256] and its posterior improvement SqueezeSegV2 [257], based on SqueezeNet architecture [109], show that very efficient semantic segmentation can be done through this projection. The more recent work from Miloto et al. [169] combines the DarkNet architecture [198] with a GPU based post-processing method for real-time semantic segmentation.

Projection-based approaches tend to be faster than other representations, but they lose the potential of learning 3D features. LuNet [26] is a recent work which proposes to learn local features using point-based operations before projecting into the 2D space. Our novel projection module tackles with this issue by including a context feature extractor based on point-based operations. Besides, we build a faster and more parameter-efficient architecture and a faster implementation of LuNet’s neighbor search method.

5.1.2 3D-MiniNet: LiDAR point cloud segmentation

Our novel approach for LIDAR semantic segmentation is summarized in Fig. 5.2. It consists of three modules: (A) fast 3D point neighbor search, (B) 3D-MiniNet, which takes P groups of N points and outputs the segmented point cloud and, (C) the KNN-based post-processing which refines the final segmentation.

There are two main issues that typically prevent point-based models to run at an acceptable frame-rate compared to projection-based methods: 3D point neighbor search is a required, but slow, operation and performing 3D operations is slower than using 2D convolutions. In order to alleviate these two issues, our approach includes a fast point neighbor search proxy (subsection 5.1.2.1), and a module to minimize expensive

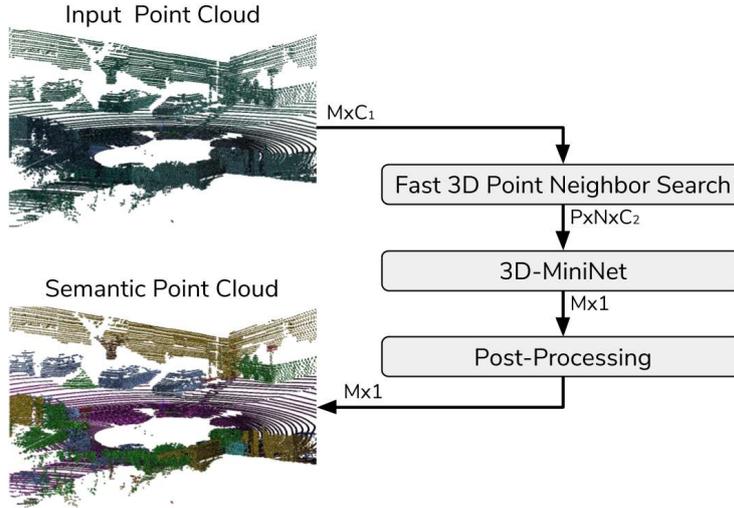


Figure 5.2: **Proposed approach overview.** The M points from the input point cloud (with C_1 features) are split into P groups of N points with our fast 3D point neighbor search. Each point has a C_1 feature vector, which is extended to C_2 in this process with data relative to each group. The proposed 3D-MiniNet takes the point groups and predicts one semantic label per point. A post-processing method [169] is used to refine the final results.

point-based operations, which takes raw 3D points and outputs a 2D representation to be processed with a 2D CNN (subsection 5.1.2.2).

5.1.2.1 Fast 3D Point Neighbor Search

We need to find the 3D neighbors because we want to learn features that encode the relationship of each point with their neighbors in order to learn information about the shape of the point-cloud. In order to perform the 3D neighbor search more efficiently, we first project the point cloud into a spherical projection of shape $W \times H$, mapping every 3D point (x, y, z) into a 2D coordinate (u, v) , i.e., $\mathbb{R}^3 \rightarrow \mathbb{R}^2$:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2} [1 - \arctan(y, x)\pi^{-1}] W \\ [1 - (\arcsin(zr^{-1}) + f_{up}) f^{-1}] H \end{pmatrix}, \quad (5.1)$$

where $f = f_{up} + f_{down}$ is the vertical field-of-view of the sensor and r is the depth of each point. We perform the projection of Eq. 5.1 following [169], where each pixel encodes one 3D point with five features: $C_1 = \{x, y, z, depth, remission\}$.

We perform the point neighbor search in the spherical projection space using a sliding-window approach. Similarly to a convolutional layer, we get groups of pixels, i.e., projected points, by sliding a $k \times k$ window across the image. The generated groups of points have no intersection, i.e., each point belongs only to one group. This step generates P point groups of N points each ($N = k^2$), where all points from the spherical projection are used ($P \times N = W \times H$).

Before feeding the actual segmentation module, 3D-MiniNet, with these point groups, the features of each point are augmented. For each group we compute the relative (r) feature values for each point. They are computed with respect to the group

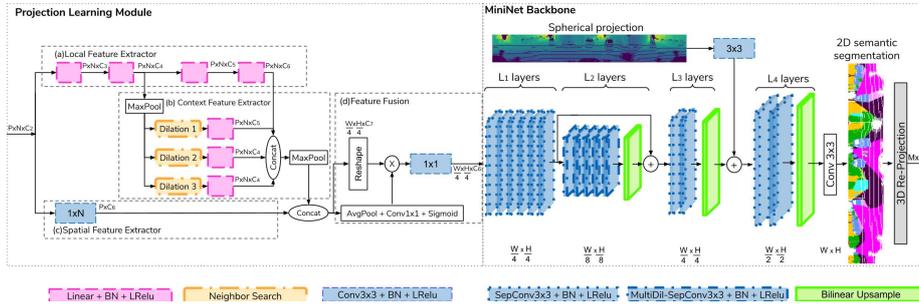


Figure 5.3: **3D-MiniNet overview.** It takes P groups of N points each and computes semantic segmentation of the M points of the point cloud where $P \times N = M$. It consists of two main modules: our proposed learning module (on the left) which learns a 2D tensor which is fed to the second module, an efficient FCNN backbone (on the right) which computes the 2D semantic segmentation. Each 3D point of the point cloud is given a semantic label based on the 2D segmentation. Best viewed in color.

mean for each C_1 feature (similar to previous works which compute features relative to a center point [25, 281]). Besides, similar to [280], we compute the 3D euclidean distance of each point to the mean point. Therefore, each point has now eleven features: $C_2 = \{x, x_r, y, y_r, z, z_r, depth, depth_r, remission, remission_r, d_{Eu}\}$.

5.1.2.2 3D-MiniNet

3D-MiniNet consists of two modules, as represented in Fig. 5.3: the proposed projection module, which takes the raw point cloud and computes a 2D representation, and our efficient backbone network based on MiniNet-v2 [11] to compute the semantic segmentation.

Projection Learning Module The goal of this module is to transform raw 3D points to a 2D representation that can be used for efficient segmentation. The input of this module is the output of the point neighbor search described in the previous subsection. It is a set of P groups, where each group contains N points with C_2 features each, gathered through the sliding-window search on the spherical projection as explained in the previous subsection.

The following three kinds of features are extracted from the input data (see left part of Fig. 5.3 for a visual description of this proposed module) and fused in the final module step:

Local Feature Extractor The first feature is a PointNet-like local feature extraction (see projection learning module (a) of Fig. 5.3). It runs four linear layers shared across the groups followed by a BatchNorm [110] and LeakyRelu [157]. We follow PointPillars [134] implementation of these shared linear layers using 1×1 convolutions across the tensor resulting in very efficient computation when handling lots of point groups.

Context Feature Extractor The second feature extraction (projection learning module (b) of Fig. 5.3) learns context information from the points. This is a very important module because although context information can be learned through the posterior

CNN, point-based operations learn different features than convolutions. Therefore, this module helps learning a richer representation with information that might not be learned through the CNN.

The input of this context feature extractor is the output of the second linear layer of the local feature extractor (giving the last linear layer as input would drop significantly the frame-rate due to the high number of features). This tensor is maxpooled (in order to complete the PointNet-like operation which work on unordered points) and then, our fast neighbor search is run to get point groups. In this case, three different groupings (using our point neighbor search) are performed with a 3×3 sliding window with different dilation rates of 1, 2, 3 respectively. Dilation rates, as in convolutional kernels [265], keep the number of grouped points low while increasing the receptive field allowing a faster context learning. We use zero-padding and a stride of 1 for keeping the same size. After every grouping we perform a linear, BatchNorm and LeakyRelu. The outputs of these two feature extractor modules are concatenated and applied a maxpool operation over the N dimension. This maxpool operation keeps the feature with higher response along the neighbor dimension, being order-invariant with respect to the neighbor dimension. The maxpool operation also makes the learning robust to pixels with no point information (spherical projection coordinates with no point projected).

Spatial Feature Extractor The last feature extraction operation is a convolutional layer of kernel $1 \times N$ (projection learning module (c) of Fig. 5.3). Convolutions can extract features of each point with respect to the neighbors when there is an underlying spatial structure which is the case, as the point groups are extracted from a 2D spherical projection. In the experiment section, we take this feature extractor as our baseline without the two others which is equivalent of performing only standard convolutions on the spherical projection.

Feature Fusion Lastly, a feature fusion with self-attention module is applied. It learns to reduce the feature space into an specified number of features, learning which features are more important. It consists of three stages: (1) concatenation of the feature extraction outputs reshaping the resulting tensor to $(W/4 \times H/4 \times C_7)$, (2) a self-attention operation which multiplies the reshaped tensor by the output of a pooling, 1×1 convolution and sigmoid function which has the same concatenated tensor as its input and, (3) a 1×1 convolutional layer followed by a BatchNorm and LeakyRelu which acts as a bottleneck limiting the output to C_6 features.

All implementation details, such as the number of features of each layer, are specified in Sect. 5.1.3. The experiments in Sect. 5.1.4 show how each part of this learning module contributes to improve 3D-MiniNet’s performance.

2D Segmentation Module (MiniNet Backbone) Once the previous module has computed a $W/4 \times H/4 \times C_6$ tensor, the 2D semantic segmentation is obtained running an efficient CNN (see MiniNet backbone in Fig. 5.3 for a visual description). Our module uses a FCNN instead of performing more MLP operations because convolutional layers have lower inference time when working on high dimensional spaces. Our FCNN is based on MiniNet-v2 architecture [11]. Our encoder performs L_1 depthwise separable convolutions and L_2 multi-dilation depthwise separable convolutions. For the decoder, we use bilinear interpolations as upsampling layers. It performs L_3 depthwise separable convolutions at $W/4 \times H/4$ resolution and L_4 at $W/2 \times H/2$ resolution. Finally,

a convolution is performed at $W \times H$ resolution to get the 2D semantic segmentation prediction.

Similarly to MiniNet-v2, we also include a second convolutional branch to extract fine-grained information, i.e., high-resolution low-level features. The input of this second branch is the spherical projection. The number of layers and features at each layer is specified in Sect. 5.1.3.2.

As a final step, the predicted 2D semantic segmentation labels are re-projected back into the 3D space ($\mathbb{R}^2 \rightarrow \mathbb{R}^3$). For the points projected into the spherical representation, this re-projection is a straightforward step, as it just implies assigning the semantic label predicted in the spherical projection. Nevertheless, the points that had not been projected into the spherical projection (one 2D coordinate can have more than one 3D point) have no semantic label. For these points, the semantic label of its corresponding 2D coordinate is assigned. As this issue may lead to miss-predictions, a post-processing method is performed to refine the results.

5.1.2.3 Post-Processing

In order to cope with the miss-predictions of non-projected 3D points, we follow Milioto et al. [169] post-processing method. All 3D points get a new semantic label based on K Nearest Neighbors (KNN). The criteria for selecting the K nearest points is not based on the relative euclidean distances but on relative depth values. Besides, the search is narrowed down based on 2D spherical coordinate distances. Milioto et al. implementation is GPU-based and is able to run in 7ms keeping the frame-rate high.

5.1.3 Experimental setup

This section details the setup used in our experimental evaluation.

5.1.3.1 Datasets

SemanticKITTI Benchmark The SemanticKITTI dataset [19] is a recent large-scale dataset that provides dense point-wise annotations for the entire KITTI Odometry Benchmark [75]. The dataset consists of over 43000 scans from which over 21000 are available for training (sequences 00 to 10) and the rest (sequences 11 to 21) are used as test set. The dataset distinguishes 22 different semantic classes from which 19 classes are evaluated on the test set via the official online platform of the benchmark. As this is the current most relevant and largest dataset of single-scan 3D LIDAR semantic segmentation, we perform our ablation study and our more thorough evaluation on this dataset.

KITTI Benchmark SqueezeSeg [256] work provided semantic segmentation labels exported from the 3D object detection challenge of the KITTI dataset [75]. It is a medium-size dataset split into 8057 training scans and 2791 validation scans.

5.1.3.2 Settings

3D Point Neighbor Search Parameters We set the resolution of the spherical projection to 2048×64 for the SemanticKITTI dataset and 512×64 for the KITTI (same resolution than previous works to be able to make fair comparisons). We set a 4×4 window size with a stride of 4 and no zero-padding for our fast point neighbor search

leading to 8192 groups of 3D points for the SemanticKITTI data and 2048 groups for the KITTI data. Our projection module is fed with these groups and generates a learned representation of resolution 512×16 for the SemanticKITTI configuration and 128×16 for the KITTI.

Network Parameters The full architecture and all its parameters are described in Fig. 5.3. We considered three different configurations for evaluating the proposed approach: 3D-MiniNet, 3D-MiniNet-small, 3D-MiniNet-tiny. The number of features (C_3, C_4, C_5, C_6) for the projection module of the different 3D-MiniNet configurations are (24, 48, 96, 192) features for 3D-MiniNet, (16, 32, 64, 128) for 3D-MiniNet-small and (12, 24, 48, 96) for 3D-MiniNet-tiny. The number of layers (L_1, L_2, L_3, L_4) of the FCNN backbone network are (50, 30, 4, 2) features for 3D-MiniNet, (24, 20, 2, 1) for 3D-MiniNet-saml and (14, 10, 2, 1) for 3D-MiniNet-tiny. N_c is the number of semantic classes of the dataset.

Post-processing Parameters For the K Nearest Neighbors post-process method [169], we set as 7×7 the windows size of the neighbor search on the 2D segmentation and we set K to 7.

Training protocol We train the different 3D-MiniNet configurations for 500 epochs with batch size of 3, 6 and 8 for 3D-MiniNet, 3D-MiniNet-small, and 3D-MiniNet-tiny respectively (different due to memory constraints). We use Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of $4 \cdot 10^{-3}$ and a decay of 0.99 every epoch. For the optimization, we use the cross-entropy loss function, see eq. 5.2.

$$\mathcal{L} = -\frac{1}{M} \sum_{m=1}^M \sum_{c=1}^C \left(\frac{f_t}{f_c}\right)^i y_{c,m} \ln(\hat{y}_{c,m}), \quad (5.2)$$

where M is the number of labeled points and C is the number of classes. $Y_{c,m}$ is a binary indicator (0 or 1) of point m belonging to a certain class c and $\hat{y}_{c,m}$ is the CNN predicted probability of point m belonging to a certain class c . This probability is calculated by applying the soft-max function to the networks' output. To account for class imbalance, we use the median frequency class balancing, as applied in SegNet [16]. To smooth the resulting class weights, we propose to apply a power operation, $w_c = \left(\frac{f_t}{f_c}\right)^i$, with f_c being the frequency of class c and f_t the median of all frequencies. We set i to 0.25.

Data augmentation During the training, we randomly rotate and shift the whole 3D point cloud. We randomly invert the sign for X and Z values for all the point cloud. We also drop some points. The rotation angle is a Gaussian distribution with mean 0 and standard deviation (std) of 40° . The shifts we perform are Gaussian distributions with mean 0 and std of 0.35, 0.35 and 0.01 (meters) for the X, Y, Z axis (being Z the height). The percentage of dropped points is a uniform distribution between 0 and 10.

5.1.4 Results

5.1.4.1 Ablation Study of the Projection Module

The projection module is the main novelty from our approach. This subsection shows how each part helps to improve the learned representation. For this experiment, we use

Table 5.1: Ablation study of the different parts of the projection module evaluated on the test set of SemantiKITTI.

Method	Data Aug.	Conv	Local MLP	Attention	Context MLP	Relative features	mIoU	FPS	Params (M)
		✓					44.4	73	0.93
	✓	✓					47.6	73	0.93
	✓		✓				48.7	69	0.93
3D-MiniNet	✓	✓	✓				49.5	66	0.96
Small	✓	✓	✓	✓			49.9	65	1.08
	✓	✓	✓	✓	✓		51.2	61	1.13
	✓	✓	✓	✓	✓	✓	51.8	61	1.13

3D-MiniNet-small configuration.

Table 5.1 shows the ablation study of our proposed module, measuring the mIoU, speed and learning parameters needed with each configuration. The first row and baseline is working on the spherical projection using a convolution as the *projection* method, i.e., just a downsampling in that case.

As the projection used is neither rotation nor shift invariant, performing this data augmentation helps to our network generalization as first row shows. Second row shows the performance using only $1 \times N$ convolutions in the learning layers with the 5-channel input (C_1) used in RangeNet [169] which we establish as our baseline, i.e., our spatial feature extractor. The third row shows the performance if we replace the $1 \times N$ convolution for point-based operations, i.e., our local feature extractor. These results point that MLP operations work better for 3D points but take more execution time. The fourth row combines both the convolution and local MLP operation. Combining convolutions and MLP operations increases performance due to the different type of features learned by each type of operation as explained in Sect. 5.1.2.2.

The attention module also increases the performance with almost no extra computational effort. It reduces the feature space into a specified number of features, learning which features are more important. The sixth row shows the results adding our context feature extractor. Context is also learned later through the FCNN via convolutions but here, the context feature extractor learns different context through with MLP operations. Context information is often very useful in semantic tasks, e.g., for distinguishing between a bicyclist, a cyclist and a motorcyclist. This context information gives a boost higher than the other feature extractors showing its relevance. Finally, increasing the number of features of each point with features relative to the point group (C_2) also leads to better performance without decreasing the frame-rate and without adding any learning parameter.

5.1.4.2 Benchmarks results

This subsection presents quantitative and qualitative results of 3D-MiniNet and comparisons with other relevant works.

Quantitative Analysis Table 5.2 compares our method with several point-based approaches (rows 1-4), 3D representation methods (row 5) and projection-based approaches (rows 6-11) measuring the mIoU, the processing speed (FPS) and the number of parameters required by each method. As we can see, point-based methods for semantic segmentation of LIDAR scans tend to be slower than projection ones without

Table 5.2: Results on single-scan test set in SemanticKITTI [19]. Point-based methods: rows 1-4. 3D representations: row 5. Projection-based methods: rows 6-11.

Methods	mIoU	Frame-rate (FPS)	Params(M)
PointNet [190]	14.6	2	3
SPG [133]	17.4	0.2	0.25
PointNet++ [191]	20.1	0.1	6
RandLA-Net [98]	53.9	22	1.24
SPLATNet [224]	18.4	1	0.8
SqueezeSeg [256]	29.5	90	1
DBLiDARNet [58]	37.6	—	2.8
SqueezeSegV2 [257]	39.7	83	1
TangentConv [232]	40.9	0.3	0.4
RangeNet21 [169]	47.4	25	25
RangeNet53 [169]	49.9	13	50
RangeNet53-KNN [169]	52.2	12	50
3D-MiniNet-tiny (Ours)	46.9	98	0.44
3D-MiniNet-small (Ours)	51.8	61	1.13
3D-MiniNet (Ours)	53.0	36	3.97
3D-MiniNet-tiny-KNN (Ours)	49.0	55	0.44
3D-MiniNet-small-KNN (Ours)	54.4	40	1.13
3D-MiniNet-KNN (Ours)	55.8	28	3.97

Scans per second have been measured using a Nvidia gtx 2080ti

— Not reported by the authors.

providing better performance. As LIDAR sensors such as Velodyne usually work at 5-20 FPS, only RandLA-Net and projection-based approaches are currently able to process in real time the full amount of data made available by the sensor.

Looking at the different configurations of 3D-MiniNet, it gets state-of-the-art using fewer parameters and being faster (3D-MiniNet-small-KNN) beating both RandLANet (point-based method), SPLATNet (3D representation) and RangeNet53-KNN (projection-based). Besides, 3D-MiniNet-KNN configuration is able to get even better performance although it needs more parameters than RandLANet. If efficiency can be traded off for performance, smaller versions of Mininet also obtain better performance metrics at higher frame-rates. 3D-MiniNet-tiny is able to run at 98 fps and, with only a 9% drop in mIoU (46.9% compared to the 29% of SqueezeSeg version that runs at 90 fps).

The post-processing method applied [169] shows its effectiveness improving the results the same way it improved RangeNet. This step is crucial to correctly process points that were not included in the spherical projection, as discussed in more detail in Sect. 5.1.2.

The scans of the KITTI dataset [75] have a lower resolution (64x512) as we can see in the evaluation reported in Table 5.3. 3D-MiniNet also gets state-of-the-art performance on LIDAR semantic segmentation on this dataset. Our approach gets considerably better performance than SqueezeSeg versions (+10-20 mIoU). 3D-MiniNet also gets better performance than LuNet and DBLiDARNet which were the previous best methods on this dataset.

Table 5.3: Results on KITTI [75] validation set.

Methods	mIoU	Frame-rate (fps)	Params(M)	car IoU	pedestrian IoU	cyclist IoU
SqueezeSeg [256]	37.2	227	1	64.6	21.8	25.1
PointSeg [251]	39.7	160	—	67.4	19.2	32.7
SqueezeSegv2 [257]	44.9	143	1	73.2	27.8	33.6
LuNet [25]	55.4	67*	23.4	72.7	46.9	46.5
DBLiDARNet [58]	56.0	—	2.8	75.1	47.4	45.4
3D-MiniNet-tiny (Ours)	45.5	245	0.44	69.6	37.5	29.5
3D-MiniNet-small (Ours)	50.6	161	1.13	74.4	40.7	36.7
3D-MiniNet (Ours)	58.0	92	3.97	75.5	49.6	48.9

Scans per second have been measured using a Nvidia gtx 2080ti

* Offline neighboring point search is not taken into account.

— Not reported by the authors.

Note that in this case, we did not evaluate the KNN post-processing since this dataset only provides 2D labels.

The experiments show that projection-based methods are more suitable for the LIDAR semantic segmentation with a good speed-performance trade-off. Besides, better results are obtained when including point-based operations to extract both context and local information from the 3D raw points into the 2D projection.

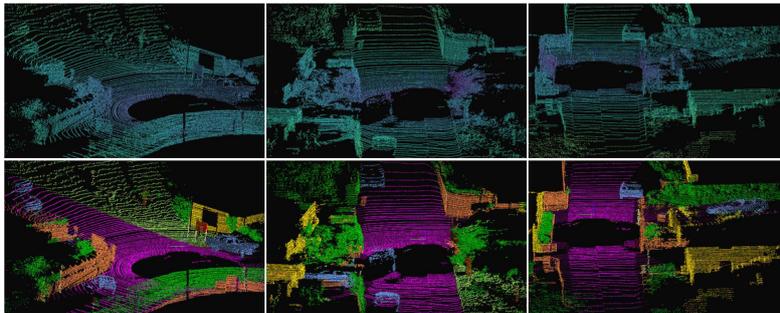


Figure 5.4: **3D-MiniNet LIDAR semantic segmentation predictions on the SemanticKITTI benchmark (test sequence 11)**. LIDAR point cloud are on top where color represents depth. Predictions are on bottom where color represents semantic classes: cars in blue, road in purple, vegetation in green, fence in orange, building in yellow and traffic sign in red. For the full video sequence, go to <https://www.youtube.com/watch?v=5ozNkgFQmSM>. Best viewed in color.

Qualitative Analysis Fig. 5.4 shows a few examples of 3D-MiniNet inference on test data. The supplementary video includes inference results on a full sequence². As test ground-truth is not provided for the test set (evaluation is performed externally on the online platform), we can only show visual results with no label comparison.

Note the high quality results on our method in relevant classes such as cars, as well as in challenging classes such as traffic signs. In the supplementary video we can also appreciate some of the 3D-MiniNet failure cases. As it could be expected, the biggest difficulties happen distinguishing between classes with similar geometric shapes and structures like building and fences.

5.1.5 Conclusions

In this work, we propose 3D-MiniNet, a fast and efficient approach for 3D LIDAR semantic segmentation. 3D-MiniNet projects the 3D point cloud into a 2-Dimensional space and then learns the semantic segmentation using a fully convolutional neural network. Differently from common projection-based approaches that perform a predefined projection, 3D-MiniNet learns this projection from the raw 3D points, learning both local and context information from point-based operations, showing very promising and effective results. Our ablation study shows how each part of the proposed approach contributes to the learning of the representation. We validate our approach on the SemanticKITTI and KITTI public benchmarks. 3D-MiniNet gets state-of-the-art results while being faster and more efficient than previous methods.

5.2 Domain Adaptation in LiDAR Semantic Segmentation

3D semantic segmentation has a wide range of applications in robotics since most autonomous systems require an accurate and robust perception of their environment. A commonly used sensor for 3D perception in robotics is the LiDAR (Light Detection And Ranging). It provides accurate distance measurements of the surrounding 3D space. In recent years, deep learning approaches are achieving state-of-the-art performance in the 3D LiDAR semantic segmentation task [9, 169]. However, deep learning methods require large amounts of labeled data to achieve high performances. Besides, deep neural networks often fail at generalizing the learned knowledge to new domains or environments. Therefore, when applying existing models on data with a different distribution than the training data, i.e., from a different domain, the performance is considerably degraded. A slight change in the data distribution can significantly drop the performance.

Domain adaptation techniques aim to eliminate or reduce this drop. Existing works for domain adaptation in semantic segmentation focus on RGB data [38, 142, 247, 293]. Most of them try to minimize the distribution shift between two different domains. Very few approaches have tackled this problem with LiDAR data [257], which equally suffers from the domain shift. RGB data commonly suffers from variations due to light and weather conditions, while the most common variations within 3D point clouds data come from sensor resolution (i.e., sensors with more laser sweeps generate denser point clouds) and from the sensor placement (because point clouds have relative coordinates with respect to the sensor). Both sensor resolution and placement issues are common

²<https://www.youtube.com/watch?v=5ozNkgFQmSM>

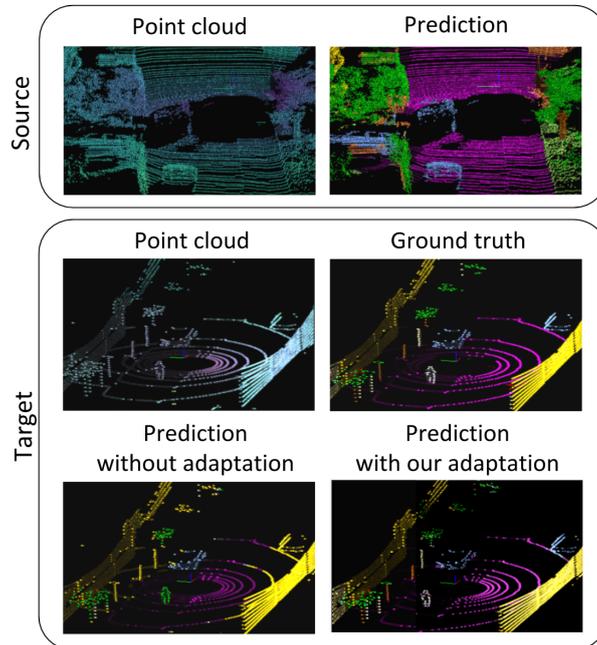


Figure 5.5: **Adaptation results of our method.** Result of our proposed approach for Domain Adaptation in LiDAR Semantic Segmentation. Given a model trained on the source domain, top row shows the result on the source domain (SemanticKitti [19]). Meanwhile, bottom row shows the result on the target domain without adaptation and the improved result applying our proposed adaptation.

examples that change the data distribution of the captured 3D point clouds. Coping with these issues would enable the use of large existing labeled LiDAR datasets for more realistic use-cases in robotic applications, reducing the need for data labeling.

This work proposes two strategies to improve unsupervised domain adaptation (UDA) in LiDAR semantic segmentation, see a sample result on Fig. 5.5. The first strategy addresses this problem by applying a set of simple steps to align the data distribution reducing the domain gap on the input space. The second strategy proposes how to align the distribution on the output space by aligning the class distribution. These two proposed strategies can be applied in conjunction with current state-of-the-art approaches boosting their performance. Our main contributions can be summarized as follows:

- Simple data processing steps (*data alignment*) that considerably help reducing the domain gap. Our results show that this step is crucial for a proper domain adaptation.
- A novel learning method for aligning the target class distribution to the source class distribution (*class alignment*) which further improves the adaptation.

We validate our approach on three different scenarios getting state-of-the-art results. We use the SemanticKitti dataset [19] as the source domain and we adapt it to SemanticPois [185], to Paris-Lille-3D [208] and to a new collected and released dataset.

5.2.1 Related Work

5.2.1.1 3D LiDAR Point Cloud Segmentation

Semantic segmentation of 3D LiDAR data aims to assign a semantic label to every point scanned by the LiDAR sensor.

Before the current trend and wide adoption of deep learning approaches, earlier methods relied on exploiting prior knowledge and geometric constraints [262]. As far as deep learning methods are concerned, there are two main types of approaches to tackle the 3D LiDAR semantic segmentation problem. On one hand, there are approaches that work directly on the 3D points, i.e., the raw point cloud is taken as the input [133, 190, 191]. On the other hand, other approaches convert this 3D point cloud into another representation (images [9], voxels [287], lattices [207]) in order to have a structured input. For LiDAR semantic segmentation, the most commonly used representation is the spherical projection [9, 25, 169, 256, 257]. Milioto et al. [169] show that point-based methods, i.e., approaches that work directly on the 3D points, are slower and tend to be less accurate than methods which project the 3D point cloud into a 2D representation and make use of convolutional layers. SqueezeSeg [256] is one of the first works that uses the spherical projection for LiDAR semantic segmentation making use of a Convolutional Neural Network (CNN). Later works have improved this approach using more complex CNNs and adding modules to the SqueezeSeg pipeline. RangeNet [169] proposes a post-processing method for improving the re-projection of the 2D resulting segmentation back to the 3D points. 3D-MiniNet [9] proposes a learning module before the CNN which takes the raw point cloud as the input and outputs a learned 2D representation.

5.2.1.2 Unsupervised Domain Adaptation for Semantic Segmentation

Unsupervised Domain Adaptation (UDA) aims to adapt models that have been trained on one specific domain (source domain) to be able to work on a different domain (target domain) where there is a certain lack of labeled training data. Most works follow similar ideas: the input data or features from a source-domain sample and a target-domain sample should be indistinguishable. Several works follow an adversarial training scheme to minimize the distribution shift between the target and source domains data. This approach has been shown to work properly at pixel space [266], at feature space [96] and at output space [247]. However, adversarial training schemes tend to present convergence problems. Alternatively, other works follow different schemes. Entropy minimization methods [38, 247] do not require complex training schemes. They rely on a loss function that minimizes the entropy of the unlabeled target domain output probabilities. This entropy minimization is closely linked to self-training methods. For self-training, pseudo-labels are generated from the unlabeled target domain output probabilities for a later training with some supervised loss such as the softmax cross-entropy [142, 293]. These self-supervised works follow an iterative and cyclic scheme where pseudo-labels help the model to improve and, as the model improves, the generated pseudo-labels present higher quality.

Regarding segmentation on LiDAR data, very few works have studied the problem of domain adaptation. SqueezeSegV2 [257] based the adaptation on existing adaptation works like correlation alignment [176]. A very recent work, Xmuda [111] focuses on combining the LiDAR information with the RGB images for multi-modal domain adaptation. They propose to apply the KL divergence between the output probabilities of both modalities as the main loss function. Besides, they also apply previously

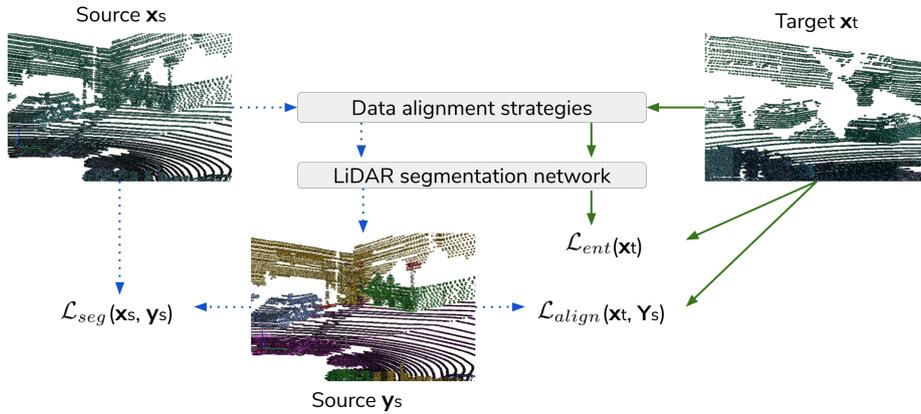


Figure 5.6: **Approach overview.** The figure shows our pipeline steps and optimization losses. First, we perform distribution alignment on the input space, i.e., data alignment strategies. Then, we optimize the segmentation loss for source samples where the labels are known and the class alignment and entropy losses for target data where no labels are available (See Sect. 5.2.2 for details). **Green continuous arrows** are used for target data and **blue pointed arrows** for source data.

proposed methods like entropy minimization [247].

This work investigates different UDA strategies (both existing and novel) to improve UDA for the particular case of LiDAR semantic segmentation. The presented results show their effectiveness in reducing the domain gap.

5.2.2 Unsupervised Domain Adaptation For LiDAR Semantic Segmentation

This section describes the proposed domain adaptation approach, including the LiDAR semantic segmentation method used, the strategies proposed to reduce the domain gap (data alignment and class distribution alignment), and the formulation of the proposed learning task. Figure 5.6 presents an overview of our proposed approach which is further explained in the following subsections.

5.2.2.1 LiDAR Semantic Segmentation Model

We use a recent method for LiDAR semantic segmentation which achieves state-of-the-art performance on several LiDAR segmentation datasets, 3D-MiniNet [9]. This method consists of three main steps. First, it learns a 2D representation from the 3D points. Then, this representation is fed to a 2D Fully Convolutional Neural Network that produces a 2D semantic segmentation. These 2D semantic labels are re-projected back to the 3D space and enhanced through a post-processing module.

Let $\mathcal{X}_s \subset \mathbb{R}^{N \times 3}$ be a set of source domain LiDAR point clouds along with associated semantic labels, $\mathcal{Y}_s \subset (1, C)^N$. Sample x_s is a point cloud of size N and $y_s^{(n,c)}$ provides the label of point (n) as one-hot encoded vector. Let F be our LiDAR segmentation network which takes a point cloud x and predicts a probability distribution (size C classes) for each point of the point-cloud $F(x) = P_x^{(n,c)}$.

The parameters θ_F of F are optimized to minimize cross-entropy loss $\mathcal{L}_{seg}(\mathbf{x}_s, \mathbf{y}_s) = -\sum_{n=1}^N \sum_{c=1}^C \mathbf{y}_s^{(n,c)} \log \mathbf{P}_{\mathbf{x}_s}^{(n,c)}$ on source domain samples. Therefore, as the supervised semantic segmentation is concerned, the optimization problem simply reads:

$$\min_{\theta_F} \frac{1}{|\mathcal{X}_s|} \sum_{\mathbf{x}_s \in \mathcal{X}_s} \mathcal{L}_{seg}(\mathbf{x}_s, \mathbf{y}_s). \quad (5.3)$$

5.2.2.2 Data alignment strategies for LiDAR

The problem of domain adaptation, i.e., data distribution misalignment, between \mathcal{X}_s and \mathcal{X}_t (a set of target domain LiDAR point clouds), can be handled on the network weights θ_F but also modifying \mathcal{X}_s and \mathcal{X}_t in order to align the distributions at the input space.

Next, we describe the different strategies for better data alignment that we propose to improve LiDAR domain adaptation.

XYZ-shift augmentation One of the main causes of misalignment for LiDAR point clouds are the changes in the vehicle set-up: the placement of the sensor in different vehicles and different locations. Since the point cloud values are relative to the sensor origin, these changes cause variations affecting the whole point cloud. Performing strong data augmentation on \mathcal{X}_s is crucial to reduce this domain gap. In this work, we perform XYZ shifts large enough to cover the different sensor set-ups. We propose to perform shifts up to ± 2 meters on the Z-axis (height) and up to ± 5 meters on the Y-axis and X-axis.

Per-class augmentation Apart from performing standard data augmentation on the whole point cloud, we also propose to perform the augmentation independently per class, in order to enrich the spatial distribution. In particular, in this work, we perform shifts up to ± 1 meters on the Z-axis (height) and up to ± 3 meters on the Y-axis and X-axis.

Same number of beams Different LiDAR sensors capture the environment differently. Besides the sensor placement and orientation, a significant difference between sensors is the number of captured beams, which results in a more sparse or dense point cloud. We propose to match the data beams between the two domains by reducing the data from the sensor with a higher number of beams ending up with more homogeneous data within \mathcal{X}_s and \mathcal{X}_t .

Only relative features Point-cloud segmentation methods commonly use both relative and absolute values of the input data for learning the segmentation. In order to be independent of absolute coordinates that are less robust compared to relative coordinates, we propose to use only relative features of the data. Therefore, for XYZ values (it can be extended for reflectance or depth), we propose to use only relative distances of every point with respect to their neighbors.

5.2.2.3 Class distribution alignment

The domain shift appears due to many different factors. For example, different environments can present quite different appearances, the spatial distribution of objects may

vary, the capturing set-up for different scenarios can be totally different, etc. Depending on the problem tackled and prior knowledge, we can hypothesize which of these differences can be neglected and assumed not to affect to the models we are learning. In this work, all the datasets used are from urban scenarios. Taking this into account, although the data distribution changes between the datasets, we can assume that the class distribution is going to be very similar across these datasets. For example, we can assume that if \mathbf{y}_s has a distribution of 90% road pixels and 10% car pixels, then \mathbf{y}_t will likely present a similar distribution.

Our approach learns parameters θ_F of F in such a way that the predicted class distribution $F(\mathcal{X}_t)$ matches the real class distribution of \mathcal{Y}_s , i.e., the histogram representing the frequency of occurrence of each class, previously computed in an offline fashion. To do so, We propose to compute the KL-divergence between these two class distribution as the class alignment loss $\mathcal{L}_{align}(\mathbf{x}_t, \mathcal{Y}_s) = \sum_{n=1}^N \mathbf{hist}(\mathcal{Y}_s) \log \frac{\mathbf{hist}(\mathcal{Y}_s)}{\mathbf{P}_{\mathbf{x}_t}^{(n)}}$. Therefore, the optimization problem reads as:

$$\min_{\theta_F} \frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_t \in \mathcal{X}_t} \mathcal{L}_{align}(\mathbf{x}_t, \mathcal{Y}_s). \quad (5.4)$$

Equation 2 requires to compute the class distribution $\mathbf{P}_{\mathbf{x}_t}$ over the whole dataset. As this is computationally unfeasible, we compute it over the batch as an approximation.

5.2.2.4 Optimization Formulation

The entropy loss is computed as in prior work MinEnt [247]:

$$\mathcal{L}_{ent}(\mathbf{x}_t) = \frac{-1}{\log(C)} \sum_{n=1}^N \sum_{c=1}^C \mathbf{P}_{\mathbf{x}_t}^{(n,c)} \log \mathbf{P}_{\mathbf{x}_t}^{(n,c)}, \quad (5.5)$$

while the segmentation loss \mathcal{L}_{seg} and the class alignment loss \mathcal{L}_{align} are computed as detailed in previous subsections.

During training, we jointly optimize the supervised segmentation loss \mathcal{L}_{seg} on source samples and the class alignment loss \mathcal{L}_{align} and entropy loss \mathcal{L}_{ent} on target samples. The final optimization problem is formulated as follows:

$$\begin{aligned} \min_{\theta_F} \frac{1}{|\mathcal{X}_s|} \sum_{\mathbf{x}_s} \mathcal{L}_{seg}(\mathbf{x}_s, \mathbf{y}_s) + \frac{\lambda_{align}}{|\mathcal{X}_t|} \sum_{\mathbf{x}_t} \mathcal{L}_{align}(\mathbf{x}_t, \mathcal{Y}_s) \\ + \frac{\lambda_{ent}}{|\mathcal{X}_t|} \sum_{\mathbf{x}_t} \mathcal{L}_{ent}(\mathbf{x}_t), \end{aligned} \quad (5.6)$$

with λ_{ent} and λ_{align} as the weighting factors of the alignment and entropy terms.

5.2.3 Experimental Setup

This section details the setup used in our evaluation.

5.2.3.1 Datasets

We use four different datasets for the evaluation. They were collected in four different geographical areas, with four different LiDAR sensors, and with four different set-ups. We take the well known SemanticKITTI dataset [19] as the source domain dataset and the other three datasets as target domain data.

SemanticKITTI The SemanticKITTI dataset [19] is a recent large-scale dataset that provides dense point-wise annotations for the entire KITTI Odometry Benchmark [75]. The dataset consists of over 43000 LiDAR scans from which over 21000 are available for training. The dataset distinguishes 22 different semantic classes. The capturing sensor is a Velodyne HDL-64E mounted on a car.

Paris-Lille-3D Paris-Lille-3D [208] is a medium-size dataset that provides three aggregated point clouds, which are built from continuous LiDAR scans of streets in Paris and Lille. It is collected with a tilted rear-mounted Velodyne HDL-32E placed on a vehicle. Following PolarNet work [279], we extract individual scans from the registered point clouds thanks to the scanner trajectory and points' timestamps. Each scan is made of points within +/- 100m. We take the Lille-1 point cloud for using the domain adaptation methods and Lille-2 for validation. We use the following intersecting semantic classes with the SemanticKitti: car, person, road, sidewalk, building, vegetation, pole, and traffic light. One thing to note is that this dataset only keeps points measured at a distance less than 20m and the LiDAR has an angle of 30 degrees between the axis of rotation and the horizontal. This configuration makes each scan to have a very limited field of view compared to other LiDAR setups.

SemanticPoss The SemanticPoss [185] is a medium-size dataset which contains 5 different sequences from urban scenarios providing 3000 LiDAR scans. The sensor used is a 40-line Pandora mounted on a vehicle. We take the three first sequences for applying the adaptation methods and the last two sequences for validation. We use the following intersecting semantic classes with the SemanticKitti: car, person, trunk, vegetation, traffic sign, pole, fence, building, rider, bike, and ground (which combines road and sidewalk in SemanticKitti).

I3A We have captured a small fourth dataset to test our approach also in a different scenario. In contrast to the three previous datasets, this dataset is not captured from a vehicle but from a small mobile robot (namely a TurtleBot³ platform). Therefore, the sensor is placed at a significantly lower height than in the other set-ups. The capturing sensor is the Velodyne VLP-16. This is a 16-line sensor that captures less dense point clouds compared to the other datasets, making the domain gap bigger. The dataset contains two sequences, one for training and another for validation. We use the intersecting semantic classes with the SemanticKitti: car, person, road, sidewalk, building, vegetation, trunk, pole, and traffic light.

5.2.3.2 Training Protocol

As we mentioned in Sec. 5.2.2.1, we use 3D-MiniNet [9] as the base LiDAR semantic segmentation method. In particular, we use the available 3D-MiniNet-small version because of memory issues. For computing the relative coordinates and features, we follow 3D-MiniNet approach extracting them from the N neighbors of each 3D point where N is set to 16.

For all the experiments we train this architecture for 700K iterations with a batch size of 8. We use Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of $5e - 3$ and a polynomial learning rate decay schedule with a power set to 0.9.

³<https://www.turtlebot.com/>

We set λ_{ent} to 0.001 as suggested in MinEnt [247] and λ_{align} to 0.001. We empirically noticed that the performance is very similar when these two hyper-parameters are set between $1e-5$ and $1e-2$. The two main conditions for them to work properly are: (1) be greater than 0 and, (2) do not be higher than the supervised loss.

One thing to take into account is that, as explained in 5.2.3.1, the Paris-Lille-3D has a very limited field of view. Therefore in order to make MinEnt [247] work in this dataset, we had to simulate the same field of view on the source dataset.

5.2.4 Results

This section presents the experimental validation of our approach compared to different baselines. The proposed approach achieves better results than the other baselines in the three different scenarios for unsupervised domain adaptation in LiDAR Semantic Segmentation. In all the experiments we use the SemanticKITTI dataset [19] as the source data distribution and perform the adaptation on the other three datasets.

5.2.4.1 Ablation Study

Table 5.4: Ablation study of our domain adaption pipeline for semantic segmentation. Source dataset: SemanticKITTI [19].

Target dataset	mIoU on I3A	mIoU on ParisLille	mIoU on SemanticPoss
Base model	15.9	19.2	13.4
+ XYZ-shift augmentation	25.1	28.9	16.3
+ Per-class augmentation	27.0	30.1	17.2
+ Same number of beams	42.0	35.4	18.3
+ Only relative features	47.1	—	19.0
+ MinEnt [247]	50.3	41.5	26.2
+ Class distribution alignment	52.5	42.7	27.0

— Not used because there was no performance gain.

The experiments in this subsection show how the different data alignment steps and the proposed learning losses affect the final performance of our approach. Table 5.4 summarizes the ablation study performed on three different scenarios. The results show how all the steps proposed contribute towards the final performance. The main insights observed in the ablation study are discussed next.

Performing strong *XYZ-shifts* results in a boost on the performance, meaning that the domain gap is considerably reduced. The distribution gap reduced by this step is the one caused by the fact of using different LiDAR sensor set-ups (such as different acquisition sensor height). Besides, in these autonomous driving set-ups, the distance between the car and the objects depends on how wide are the streets or on which lane is the data capturing source. Therefore, this is an essential and really easy data transformation to perform which gives an average of 7.2% MIoU gain.

The *per-class data augmentation* also boosts the performance. This data augmentation method tries to reduce the domain gap by adding different relative distances between different classes gaining an average of 1.3% MIoU gain.

Another interesting and straightforward technique to perform is to *match the number of LiDAR beams* of the source and target data, i.e., match LiDAR point-cloud resolution. This helps the data alignment especially for having the same point density on the 3D point-cloud and similar relative distances between the points. We show that this method gives an improvement similar to the XYZ-shift data augmentation, hugely reducing the domain gap. We can observe that the higher the initial difference in the number of beams, the more improvement we can get: the i3A LiDAR has 16 beams, the ParisLille 32, and the SemanticPoss 40, compared to the 64 of the source data (SemanticKitti).

The use of *relative features only* does not always help to reduce the domain gap, it was only beneficial on the i3A and SemanticPoss datasets. Removing the absolute features and only learning from relative features helps especially when the relative distances between the 3D points have less domain shift than the absolute coordinates. This will depend on the dataset, but the stronger the differences between capturing sensors, the more likely that the use of relative features will help.

Besides the data alignment steps, our approach includes two *learning losses* to the pipeline to help to reduce the domain gap. The first one is the entropy minimization loss proposed in MinEnt [247]. The second one is our proposed class distribution alignment loss introduced in this work. We show that these two losses can be combined for the domain adaptation problem and that, although less significantly with respect to previously discussed steps, they also improve on the three different set-ups, contributing to achieving state-of-the-art performance.

5.2.4.2 Comparison with other baselines

Table 5.5: Results on the three different LiDAR semantic segmentation datasets using different domain adaptation methods. The source dataset is the SemanticKitti dataset [19]

	mIoU on I3A	mIoU on ParisLille	mIoU on SemanticPoss
Baseline	15.9	19.2	13.4
MinEnt [247]	28.4	23.2	19.6
AdvEnt [247]	21.0	20.7	19.5
MaxSquare [38]	28.4	22.8	19.3
Data alignment (ours)*	47.1	36.2	19.0
Full approach (ours)	52.5	42.7	27.0

* Only data alignment strategies from Sect. 5.2.2.2

Table 5.5 shows the comparison of our pipeline (composed of all the steps discussed in the ablation study) with other existing methods for domain adaptation. We select MinEnt, Advent, and MaxSquare as the baselines because they are leading the state-of-the-art for unsupervised domain adaptation. We use the available authors' code for replication.

We apply the different domain adaptation methods of the three different set-ups without our data alignment steps. This comparison shows that good pre-processing of the data can obtain better results than just applying out-of-the-box methods for domain adaptation. It also shows that our complete pipeline outperforms these previous meth-

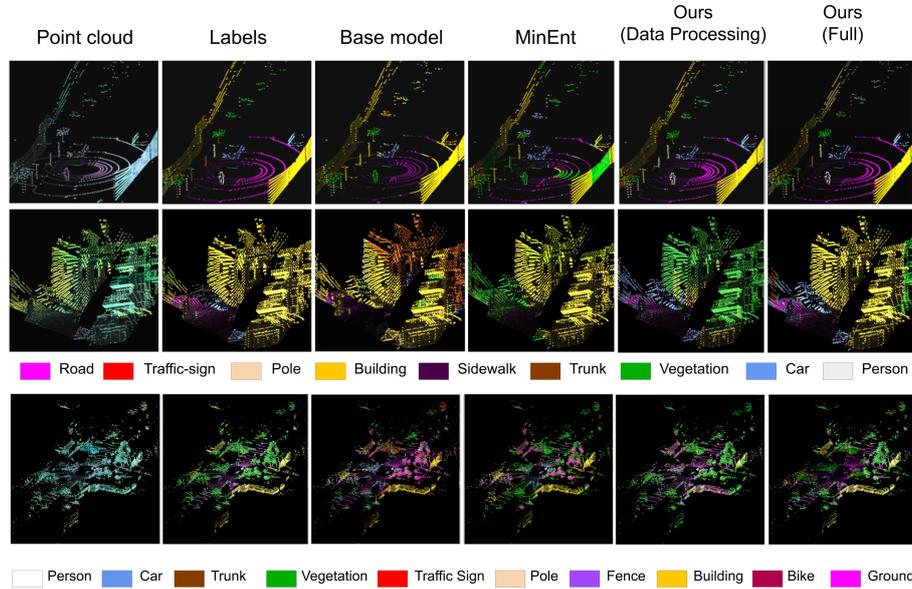


Figure 5.7: **Visual results.** Visual results of the LiDAR domain adaptation with different adaptation methods for one example from each target dataset: I3A dataset first row, ParisLille dataset second row, and SemanticPoisson last row. From left to right: Input point cloud, ground truth labels, baseline with no adaptation (trained on SemanticKitti), MinEnt [247] adaptation approach, our adaptation only with data processing strategies, our full adaptation pipeline. Best viewed in color.

ods on LiDAR domain adaptation. Our results demonstrate that combining proper data processing with learning methods for domain adaptation gives an average of more than $\times 2$ boost on the performance.

Figure 5.7 includes a few examples of the segmentation obtained with a baseline with no domain adaptation, using the MinEnt [247] approach only, with our approach using only the data pre-processing steps, and with our approach including all steps proposed. We can appreciate in figure 5.7 how data processing helps on certain semantic classes, such as road, person, car, or vegetation, while MinEnt usually improves at different ones like building. This suggests the good complementary of both strategies, and indeed combining them provides the best results. Additional results can be seen in the supplementary video.

5.2.5 Conclusions

In this work, we introduce a novel pipeline that addresses the task of unsupervised domain adaptation for LiDAR semantic segmentation. Our pipeline consists of aligning data distributions on the data space with different simple strategies combined with learning losses on the semantic segmentation process that also force the data distribution alignment.

Our results show that a proper data alignment on the input space can produce better domain adaptation results that just using out-of-the-box state-of-the-art learning methods. Besides, we show that combining these data alignment methods with learning methods, like the one proposed in this work to align the class distributions of the data,

can reduce even more the domain gap getting new state-of-the-art results. Our approach is validated on three different scenarios, from different datasets, as the target domain, where we show that our full pipeline improves previous methods on all three scenarios.

5.3 EV-SegNet: Semantic Segmentation for Event-based Cameras

Event cameras, as Dynamic Vision Sensor (DVS) [143], are promising sensors which register intensity changes of the captured environment. In contrast to conventional cameras, this sensor does not acquire images at a fixed frame-rate. These cameras, as their name suggests, capture events and record a stream of asynchronous events. An event indicates an intensity change at a specific moment and at a particular pixel (more details on how events are acquired in Section 5.3.2.2). Event cameras offer multiple advantages over more conventional cameras, 1) mainly its very high temporal resolution, which allows the capture of multiple events in microseconds; 2) its very high dynamic range, which allows the information capture at difficult lighting environments; 3) its low power and bandwidth requirements. Maqueda et al. [160] show how visual recognition tasks can benefit from these advantages in their work emphasizing that event cameras are natural motion detectors and automatically filter out any temporally-redundant information. Besides, they show that these cameras provide richer information than just subtracting consecutive conventional images.

These cameras offer a wide range of new possibilities and features that could boost solutions for many computer vision applications. However, new algorithms still have to be developed in order to fully exploit their capabilities, especially regarding recognition tasks. Most of the latest achievements based on deep learning solutions for image data have not yet been even attempted on event cameras. One of the main reasons is the output of these cameras: they do not provide standard images, and there is not yet a clearly adopted way of representing the stream of events to feed a CNN. Another challenge is the lack of labeled training data, which is key to training most recognition

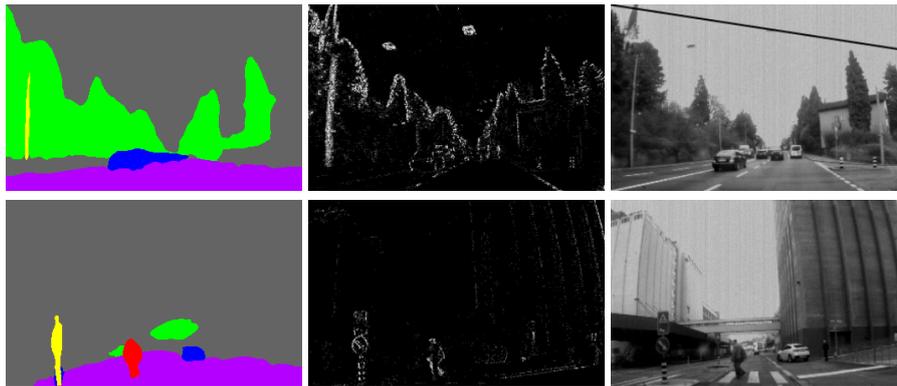


Figure 5.8: **Semantic segmentation with event-based cameras.** Two examples of semantic segmentation (left) from event based camera data (middle). The semantic segmentation is the prediction of our CNN, fed only with event data. Grayscale images (right) are displayed only to facilitate visualization.

models. Our work includes simple but effective novel ideas to deal with these two challenges. They could be helpful in many DVS applications, but we focus on an application not explored yet with this sensor, semantic segmentation.

This work proposes to combine the potential of event cameras with deep learning techniques on the challenging task of semantic segmentation. Semantic segmentation may intuitively seem a task much better suited to models using appearance information. However, we show how, with an appropriate model and representation, event cameras provide very promising results for this task.

Figure 5.8 shows two visual results as an example of the output of our work. Our main contributions are:

- First results, up to our knowledge, on semantic segmentation using DVS data. We build an Xception-based CNN that takes this data as input. Since there is no benchmark available for this problem, we propose how to generate approximated semantic segmentation labels for some sequences of the DDD17 event-based dataset. Model and data are being released.
- A comparison of different DVS data representation performance on semantic segmentation (including a new proposed representation that is shown to outperform existing ones), and an analysis of benefits and drawbacks compared to conventional images.

5.3.1 Related work

5.3.1.1 Event Camera Applications

As previously mentioned, event cameras provide valuable advantages over conventional cameras in many situations. We find recent works which have proved these advantages in several tasks typically solved with conventional vision sensors. Most of these works focus their efforts on 3D reconstruction [121, 195, 285, 290] and 6-DOF camera tracking [70, 197]. Although 3D reconstruction and localization solutions are very mature on RGB images, existing algorithms cannot be applied exactly the same way on event cameras. The aforementioned works propose different approaches for adapting them.

We find recent approaches that explore the use of these cameras for other tasks, such as optical flow estimation [71, 148, 289] or, closer to our target tasks, object detection and recognition [39, 132, 184, 219]. Regarding the data used in these recognition works, Orchard et al. [184] and Lagorce et al. [132] performed the recognition task on small datasets, detecting mainly characters and numbers. The most recent works, start to use more challenging (but scarce) recordings in real scenarios, such as N-CARS dataset, used in Sironi et al. [219], or DDD17 dataset [27], which we use in this work because of the real world urban scenarios it contains.

Most of these approaches have a common first step: encode the event information into an image-like representation, in order to facilitate its processing.

We discuss in detail different previous work event representations (encoding spatial and temporal information) as well as our proposed representation (with a different way of encoding the temporal information) in Sec. 5.3.2.

5.3.1.2 Semantic Segmentation

Semantic segmentation is a visual recognition problem which consists of assigning a semantic label to each pixel in the image. State-of-the-art on this problem is currently

achieved by deep learning based solutions, most of them proposing different variations of encoder-decoder CNN architectures [35, 36, 91, 112].

Some of the existing solutions for semantic segmentation target an instance-level semantic segmentation, e.g., Mask-RCNN [89], that includes three main steps: region proposal, binary segmentation, and classification. Other solutions, such as DeepLabv3+ [36], target class-level semantic segmentation. Deeplabv3+ is a fully convolutional extension of Xception [46], which is also a state-of-the-art architecture for image classification and the base architecture of our work. A survey on image segmentation by Zhu et al. [291] provides a detailed compilation of more conventional solutions for semantic segmentation, while Garcia et al. [73] present a discussion of more recent deep learning based approaches for semantic segmentation, covering from new architectures to common datasets.

The works discussed so far show the effectiveness of CNNs for semantic segmentation using *RGB* images. Closer to our work, we find additional works which prove great performance in semantic segmentation tasks using additional input data modalities to the standard *RGB* image. For example, a common additional input data for semantic segmentation is depth information. Cao et al. [31] and Gupta et al. [86] are two good examples of how to combine *RGB* images with depth information using convolutional neural networks. Similarly, a very common sensor in the robotics field, the LiDAR sensor, has also been shown to provide useful additional information when performing semantic segmentation [55, 229]. Other works show how to combine less frequent modalities such as fluorescence information [5] or how to perform semantic segmentation on multi-spectral images [55]. Semantic segmentation tasks for medical image analysis [145] also typically apply or adapt CNN based approaches designed for *RGB* images to different medical imaging sensors, such as MRI images [118] and CT data [47].

Our work is focused on a different modality, event camera data, not explored in prior work for semantic segmentation. Following one of the top performing models on semantic segmentation for *RGB* images [36], we base our network on the Xception design [46] to build an encoder-decoder architecture for semantic segmentation on event images. Our experiments show good semantic segmentation results using only event data from a public benchmark [27], close to what is achieved on standard imagery from the same scenarios. We also demonstrate the complementary benefits that this modality can bring when combined with standard cameras to solve this problem more accurately.

5.3.2 From Events to Semantic Segmentation

In this section, we will discuss different event representations used in visual recognition tasks in order to end up proposing a rich encoding of the event data for semantic segmentation.

5.3.2.1 Event data

Event cameras capture the changes in intensities for each pixel. The output of an event camera is not a 3-dimensional image (height, width, and channels) as conventional cameras but a stream of events. An event represents the positive or negative change in the log of the intensity signal (over an established threshold σ):

$$\log(I_{t+1}) - \log(I_t) \geq \sigma, \quad (5.7)$$

being I_{t+1} and I_t the intensity captured at two consecutive timestamps. Each event (e_i) is then defined by four different components: two coordinates (x_i, y_i) of the pixel where the change has been measured, a polarity (p_i) that can be positive or negative, and a timestamp (t_i):

$$e_i = \{x_i, y_i, p_i, t_i\}. \quad (5.8)$$

Events are asynchronous and have the described encoding that, by construction, does not provide good input for broadly used techniques in visual recognition tasks, such as CNNs. Perhaps the most straightforward representation would be a $nx4$ matrix, with n the number of events. But obviously, this representation does not encode the spatial relationship between events. Several strategies have been proposed to encode this information into a dense representation successfully applied in different applications.

5.3.2.2 Event Representation

Basic dense encoding of event location. The most successfully applied event data representation creates an image with several channels encoding the following information. It stores at each location (x_i, y_i) information from the events that happened there at any time t_i within an established integration interval of size T . Variations of this representation have been used by many previous works, showing great performance in very different applications: optical flow estimation [289], object detection [39], classification [132, 186, 219] and regression tasks [160], respectively.

Earlier works used only one channel to encode event occurrences. Nguyen et al. [182] stores the information of the last event that has occurred in each pixel, i.e., the corresponding value chosen to represent a positive event, negative event or absence of events. One important drawback is that only the last event information remains.

In a more complete representation, a recent work for steering wheel angle estimation, from Maqueda et al. [160], stores the positive and negative event occurrences into two different channels. In other words, this representation ($Hist$) encodes the 2D histogram of positive and negative events that occurred at each pixel (x_i, y_i), as follows:

$$Hist(x, y, p) = \sum_{i=1, t_i \in W}^N \delta(x_i, x) \delta(y_i, y) \delta(p_i, p), \quad (5.9)$$

where δ is the Kronecker delta function (the function is 1 if the variables are equal, and 0 otherwise), W is the time window, or interval, considered to aggregate the event information, and N is the number of events occurred within interval W . Therefore, the multiplication $\delta(x_i, x) \delta(y_i, y) \delta(p_i, p)$ denotes whether an event e_i matches its coordinates x_i, y_i with x, y values and its polarity p_i with p . This representation has two channels, one per polarity p (positive and negative events). Our proposed representation described later, will make use of these two histogram channels.

Note that all the representations discussed so far only use the temporal information (timestamps t_i) to see the time interval where each event belongs to.

Dense encodings including temporal information. However, temporal information, i.e., the timestamp of each event t_i , contains useful information for recognition tasks, and it has been shown that including this non-spatial information of each event into the image-like encodings is useful. Lagorce et al. [132] propose a 2-channel image, one channel per polarity, called *time surfaces*. They store, for each pixel, information

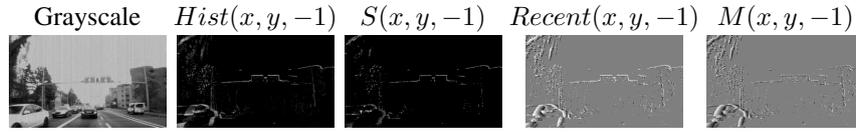


Figure 5.9: **Channels visualization.** Visualization (between 0 and 255 gray values) of different 1-channel encodings of data from events with negative polarity ($p = -1$) explained in the Sect. 5.3.2.2. In these examples the event information has been integrated for a time interval of 50ms ($T = 50ms$). Grayscale is shown as reference.

relative only to the most recent event timestamp during the integration interval W . Later, Sironi et al. [219] enhance this previous representation by changing the definition of the *time surfaces*. They now compute the value for each pixel combining information from all the timestamps of events that occurred within W .

Another recently proposed approach by Zhu et al. [289] introduces a more complete representation that includes both channels of event occurrence histograms from Maqueda et al. [160], and two more channels containing temporal information. These two channels (*Recent*) store, at each pixel (x_i, y_i) , the normalized timestamp of the most recent positive or negative event, respectively, that occurred in that location during the integration interval:

$$Recent(x, y, p) = \max_{t_i \in W} t_i \delta(x_i, x) \delta(y_i, y) \delta(p_i, p). \quad (5.10)$$

All these recent representations normalize the event timestamps and histograms to be relative values within the interval W .

Inspired by all this prior work, we propose an alternative representation that combines the best ideas demonstrated so far: the 2-channels of event histograms to encode the spatial distribution of events, together with information regarding all timestamps occurred during the integrated time interval.

Our proposed representation. We propose a 6-channel image representation. The first two channels are the histogram of positive and negative events (eq. 5.9). The remaining four channels are a simple but effective way to store information relative to all event timestamps happening during interval W . We could see it as a way to store how they are distributed along T rather than selecting just one of the timestamps. We propose to store the mean (M) and standard deviation (S) of the normalized timestamps of events happening at each pixel (x_i, y_i) , computed separately for the positive and negative events, as follows:

$$M(x, y, p) = \frac{1}{Hist(x, y, p)} \sum_{i=1, t_i \in W}^N t_i \delta(x_i, x) \delta(y_i, y) \delta(p_i, p), \quad (5.11)$$

$$S(x, y, p) = \sqrt{\frac{\sum_{i=1, t_i \in W}^N (t_i \delta(x_i, x) \delta(y_i, y) \delta(p_i, p) - Mean(x, y, p))^2}{Hist(x, y, p) - 1}}. \quad (5.12)$$

Then, our representation consists of these six 2D-channels:

$$Hist(x, y, -1), Hist(x, y, +1), M(x, y, -1), M(x, y, +1), S(x, y, -1), S(x, y, +1).$$

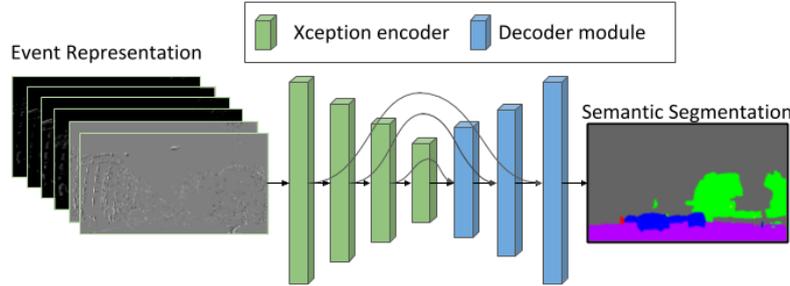


Figure 5.10: **Semantic segmentation from event based cameras.** We process the different 2D event-data encodings with our encoder-decoder architecture based on Xception [46] (Sec. 5.3.2.3 for more details). Best viewed in color.

Figure 5.9 shows a visualization of some of these channels. In the event representation images, the brighter the pixels, the higher the value encoded, e.g., white means the highest number of negative events in the $Hist(x, y, -1)$.

5.3.2.3 Semantic Segmentation from Event Data

CNNs have already been shown to work well on dense event-data representations, detailed in previous section [160, 289], therefore we explore a CNN based architecture to learn a different visual task, semantic segmentation.

Semantic segmentation is often modeled as a per-pixel classification, and therefore the output of semantic segmentation models has the same resolution that the input. As previously mentioned, there are plenty of recent successful CNN-based approaches to solve this problem both using RGB data and additional modalities. We have built an architecture inspired on current state-of-the-art semantic segmentation CNNs, slightly adapted to use the event data encodings. Related works commonly follow an encoder-decoder architecture, as we do. As the encoder, we use the well-known Xception model [46], which has been shown to outperform other encoders, both in classification [46] and semantic segmentation tasks [36]. As the decoder, also following state-of-the-art works [35, 36], we build a *light* decoder, concentrating the heavy computation on the encoder. Our architecture also includes features from the most successful recent models for semantic segmentation, including: the use of skip connections to help the optimization of deep architectures [91, 112] to avoid the vanishing gradient problem and the use of an auxiliary loss [283] which also improves the convergence of the learning process. Fig. 5.10 shows a diagram of the architecture built in this work, with the multi-channel event representation as network input.

As similar architectures, we perform the training optimization via back-propagation of the loss, calculated as the summation of all per-pixel losses, through the parameter gradients. We use the common soft-max cross entropy loss function (\mathcal{L}) described in eq. (5.13):

$$\mathcal{L} = -\frac{1}{N} \sum_{j=1}^N \sum_{c=1}^M y_{c,j} \log(\hat{y}_{c,j}), \quad (5.13)$$

where N is the number of labeled pixels and M is the number of classes. $y_{c,j}$ is a binary indicator of pixel j belonging to class c (ground truth). $\hat{y}_{c,j}$ is the CNN predicted

probability of pixel j belonging to class c .

5.3.3 Ev-Seg: Event-Segmentation Data

The Ev-Seg data is an extension for semantic segmentation of the DDD17 dataset [27]. The DDD17 dataset consists of 40 sequences of different driving set-ups. These sequences were recorded on different scenarios (e.g., motorways and urban scenarios). This dataset provides synchronized grayscale and event-based information but, it does not provide semantic segmentation labels.

Our extension includes generated (automatically generated, non-manual annotations) semantic segmentation labels to be used as ground truth for a large subset of that dataset. Besides the labels, to facilitate replication and further experimentation, together with the labeling, we also publish the selected subset of grayscale images and corresponding event data encoded with three different representations (Maqueda et al. [160], Zhu et al. [289] and the new one proposed in this work).

Generating the labels. Besides the obvious burden of manually labeling a semantic segmentation per-pixel ground truth, if we think of performing this task directly on the event-based data it turns out even more challenging. We only need to look at any of the event representations available (see Fig. 5.8), to realize that for the human eye is hard to distinguish many of the classes there if the grayscale image is not side-by-side. Other works have shown how CNNs are robust to training with noise [226] or approximated labels [5], including the work of Chen et al. [39] that also successfully uses generated labels from grayscale for object detection in event-based data. We then propose to use the corresponding grayscale images to generate an approximated set of labels for training, which we demonstrate is enough to train models to segment directly on event-based data.

To generate these approximated semantic labels, we performed the following three steps.

First, we have trained a CNN for semantic segmentation on the well known urban environment dataset *Cityscapes* [50], but using grayscale images. The architecture used for this step is the same architecture described in subsection 5.3.2.3, which follows state-of-the-art components for semantic segmentation. This grayscale segmentation model was trained for 70 epochs with a learning rate of $1e-4$. The final model obtains 83% categories MIOU on the *Cityscapes* validation data. This is still a bit far from the top results obtained on that dataset with RGB images (92% MIOU), but enough quality for our the process.

Secondly, with this grayscale model, we obtained the semantic segmentation on all grayscale images of the selected sequences (we detail next which sequences were used and why). These segmentations are what we will consider the labels to train our event-based segmentation model.

Lastly, as a final post-processing step on the ground truth labels, we cropped the bottom part of all the images, i.e., 60 bottom rows of the image it always contains the car dashboard and it only introduces noise into the generated labels.

Subset of DDD17 sequences selection. As previously mentioned, we have not generated the labels for all the DDD17 data. We next discuss the reasons and selection criteria that we followed.

Dataset Classes:	<i>flat</i> (road and pavement), <i>background</i> (construction and sky), <i>object</i> , <i>vegetation</i> , <i>human</i> , <i>vehicle</i>	
Train Sequences	Selected suitable sequence intervals	Num. Frames
1487339175	[0, 4150), [5200, 6600)	5550
1487842276	[1310, 1400), [1900, 2000), [2600, 3550)	1140
1487593224	[870, 2190)	995
1487846842	[380, 500), [1800, 2150), [2575, 2730), [3530, 3900)	1320
1487779465	[1800, 3400), [4000, 4700), [8400, 8630), [8800, 9160), [9920, 10175), [18500, 22300)	6945
	TOTAL:	15950
Test Sequences	Selected suitable sequence intervals	Num. Frames
1487417411	[100, 1500), [2150, 3100), [3200, 4430), [4840, 5150)	3890

Table 5.6: Summary of Ev-Seg Data which consists of several intervals of some sequences of the DDD17 dataset.

Some DDD17 sequences did not give good labels when being pass through the CNN. There are several reasons for this. As the data domain available to train the base grayscale semantic segmentation model was *Cityscapes* data, which is an urban domain, we selected only the DDD17 sequences from urban scenarios. Besides, only images with enough contrast (not too bright, not too dark) are likely to provide a good generated ground truth. Therefore, we only selected sequences which were recorded during day-time, with no extreme overexposure. Given these restrictions, only six sequences approximately matched them. Therefore, we performed a manual more detailed annotation of the intervals in each of these sequences where the restrictions applied (details on Table 5.6).

Data summary. Table 5.6 shows a summary of the contents of the Ev-Seg data. From the six sequences selected as detailed previously, five sequences were used as training data and one sequence was used for testing. We chose for testing the sequence with more homogeneous class distribution, i.e., that contained more amount of labels of categories which appears less such as the human/pedestrian label.

The labels have the same categories than the well-known *Cityscapes* dataset [50] (see Table 5.6), with the exception of *sky* and *construction* categories. Although these two categories were properly learned in the *Cityscapes* dataset, when performing inferences on the DDD17 dataset with grayscale images, these categories were not correctly generated due to the domain-shift. Therefore in our experiments, those two categories are learned together, as if they were the same thing. This domain shift between the *Cityscapes* and DDD17 datasets was also the cause of generating the *Cityscapes* categories instead of its classes.

Figure 5.11 shows three examples of grayscale images and corresponding generated segmentation that belong to our extension of the *DDD17 dataset*. We can see that although the labels are not as perfect as if manually annotated (and as previously mentioned, classes such as *building* and *sky* were not properly learned using only grayscale), they are pretty accurate and well defined.

5.3.4 Experimental Validation

5.3.4.1 Experiment Set-up and Metrics

Metrics. Our work addresses the semantic segmentation problem, i.e., per pixel classification, using event cameras. Thus, we evaluate our results on the standard metrics for classification and semantic segmentation: *Accuracy* and *Mean Intersection over Union (MIoU)*.

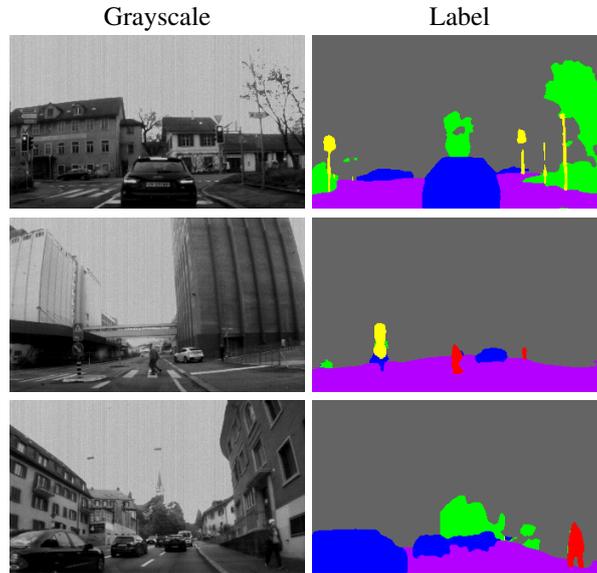


Figure 5.11: **Examples of the test sequence.** Semantic label images (right) have been generated from the grayscale images (left) through a CNN trained on a grayscale version of Cityscapes.

In semantic segmentation, given a predicted image \hat{y} and a ground-truth image y , and being N their number of pixels, which can be classified in C different classes, the accuracy metric, eq. (5.14) is computed as:

$$Accuracy(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N \delta(y_i, \hat{y}_i), \quad (5.14)$$

and the MIoU is calculated per class as:

$$MIoU(y, \hat{y}) = \frac{1}{C} \sum_{j=1}^C \frac{\sum_{i=1}^N \delta(y_{i,c}, 1) \delta(\hat{y}_{i,c}, 1)}{\sum_{i=1}^N \max(1, \delta(y_{i,c}, 1) + \delta(\hat{y}_{i,c}, 1))}, \quad (5.15)$$

where δ denotes the Kronecker delta function, y_i indicates the class where pixel i belongs to, and $y_{i,c}$ is a boolean that indicates if pixel i belongs to a certain class c .

Set-up. We perform the experiments using the CNN explained in Sec. 5.3.2.3. and the Ev-Seg data detailed in Sec. 5.3.3. We train all model variations from scratch using: the Adam optimizer with an initial learning rate of $1e-4$ and a polynomial learning rate decay schedule. We train for $30K$ iterations using a batch size of 8 and during training we perform several data augmentation steps: crops, rotations (-15° , 15°), vertical and horizontal shifts (-25% , 25%) and horizontal flips. Regarding the event information encoding, for training, we always use an integration time interval $T = 50ms$ which has been shown to perform well on this dataset [160].

5.3.4.2 Event Semantic Segmentation

Input representation	Accuracy 50ms	MIoU 50ms	Accuracy 10ms	MIoU 10ms	Accuracy 250ms	MIoU 250ms
Basic dense encoding [160]	88.85	53.07	85.06	42.93	87.09	45.66
Temporal dense encoding [289]	88.99	52.32	86.35	43.65	85.89	45.12
Ours	89.76	54.81	86.46	45.85	87.72	47.56
Grayscale	94.67	64.98	94.67	64.98	94.67	64.98
Grayscale & Ours	95.22	68.36	95.18	67.95	95.29	68.26

Table 5.7: Semantic segmentation performance of different input representations on the test Ev-Seg data. Models trained using time intervals (T) of 50ms but tested with different T values: 50ms, 10ms and 250ms.

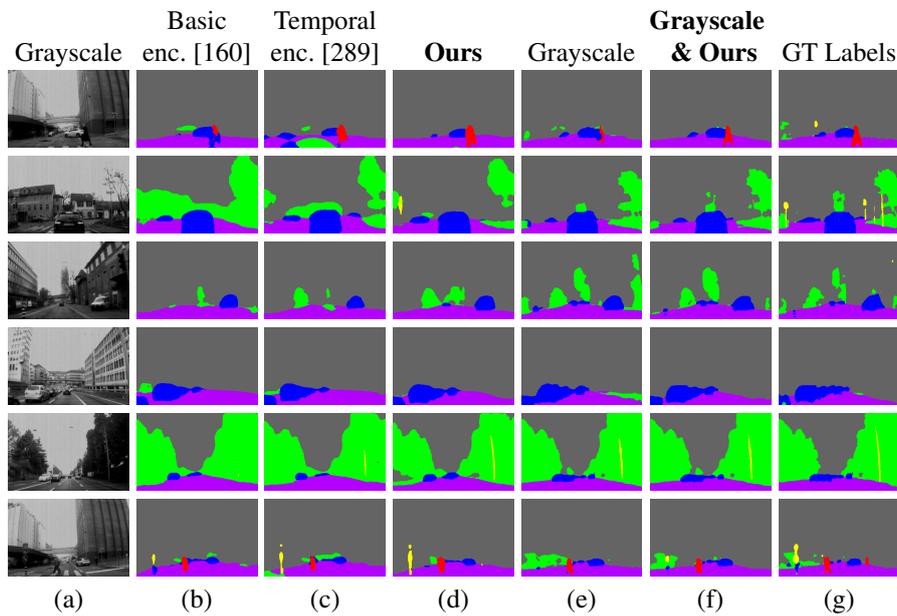


Figure 5.12: **Semantic segmentation on several test images from Ev-Seg data.** Results using different input representations of event data only, (b) to (d), or using grayscale data (e) and (f). Grayscale original image (a) and ground truth labels are shown for visualization purposes. Models trained and tested on time intervals of 50ms. Best viewed in color.

Input representation comparison. A good input representation is very important for a CNN to properly learn and exploit the input information. Table 5.7 compares several semantic segmentation models trained with different input representations. The top three rows correspond to event-based representations. We compare a basic dense encoding of event locations, a dense encoding which also includes temporal information and our proposed encoding (see Sec.5.3.2.2. for details). Our event encoding performs slightly but consistently better on the semantic segmentation task on the different metrics and evaluations considered. Fig. 5.12 shows a few visual examples of these results.

All models (same architecture, just trained with different inputs) have been trained with data encoded using integration intervals of 50ms, but we also evaluate them using

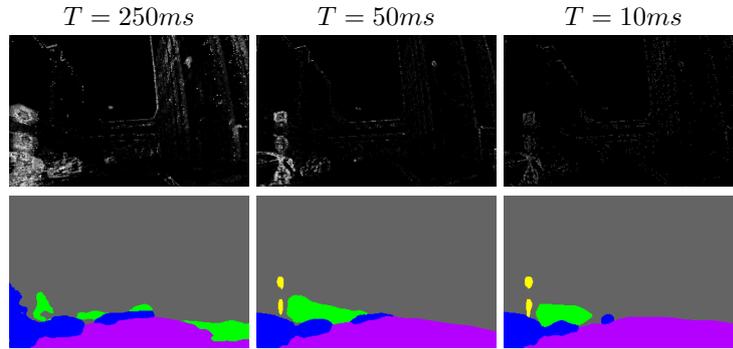


Figure 5.13: **Segmentation results.** Semantic segmentation results (bottom) using different integration interval size (T) for the event data representation (top). Results obtained with a model trained only on 50ms integrated event information encoded with our proposed representation.

different interval sizes. This is an interesting evaluation because by changing the time interval, in which the event information is aggregated, we somehow simulate different camera movement speeds. In other words, intervals of 50ms or 10ms may encode exactly the same movement but at different speeds. This point is pretty important because, in real scenarios, models have to perform well at different speeds. We can see that all models perform just slightly worse on test data encoded with different intervals sizes (10ms, 250ms) that the integration time used during training (50ms), see Fig. 5.13 examples. There are two main explanations for why the models are performing similarly on different integration intervals: 1) the encodings are normalized and 2) the training data contains different camera speeds. Both things help to generalize better at different time intervals or movement speeds.

Event vs conventional cameras. Table 5.7 also includes, in the two bottom rows, results using the corresponding grayscale image for the semantic segmentation task.

Although conventional cameras capture richer pure appearance information than event cameras, event cameras provide motion information, which is also very useful for the semantic segmentation task. In examples of results using grayscale data from Fig. 5.12(e), (f), we can see how event information helps for example to better segment moving objects, such as pedestrians (in red in those examples) or to refine object borders. While conventional cameras suffer detecting small objects and in general, with any recognition on extreme illumination (bright or dark) conditions, event cameras suffer more in recognizing objects with no movement (because they move at the same speed than the camera or because they are too far to appreciate their movement).

Conventional cameras perform better on their own for semantic segmentation than event-based cameras on their own. However, our results show that semantic segmentation results are better when combining both of them. This suggests they are learning complementary information. Interestingly, we should note that the data available for training and evaluation is precisely data where we could properly segment the grayscale image, therefore slightly more beneficial for grayscale images than event-based data (i.e., there is no night-time image included in the evaluation set because there is no ground truth for those).

Two clear complementary situations from our experiments: 1) On one hand, it is

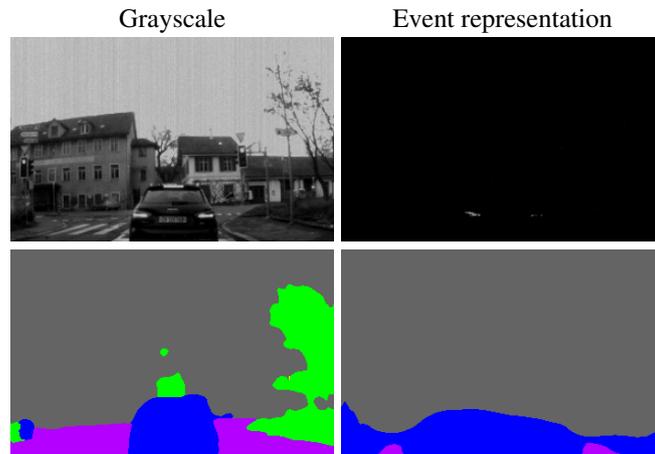


Figure 5.14: **Adversarial case results.** Semantic segmentation result (bottom) on a static sequence, i.e., a car waiting at a crossing. This is an obvious adversarial case for event cameras, due to lack of event information.

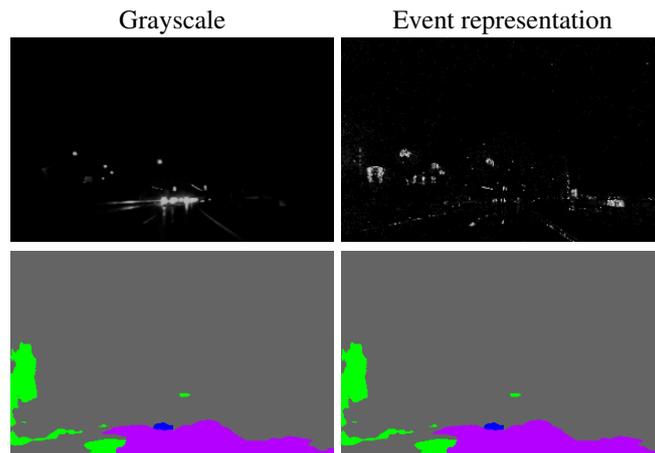


Figure 5.15: **Semantic segmentation (bottom) on extreme lighting conditions (night-time).** with different input representations (top): grayscale image and our event data representation. Corresponding models trained only on good illuminated daytime samples. This is an obvious adversarial case for conventional cameras, due to lack of information in the grayscale capture.

already known that one the major drawback of event cameras is that objects that do not move with respect to the camera do not trigger events, i.e., are invisible. Fig. 5.14 shows an example of a car waiting at a pedestrian crossing, where we see that while conventional cameras can perfectly see the whole scene, event cameras barely capture any information; 2) On the other hand, event cameras are able to capture meaningful information on situations where scene objects are not visible at all for conventional vision sensors, e.g., difficult lighting environments. This is due to their high dynamic range, Fig. 5.15 illustrates an example of a situation where neither of the grayscale nor

the event-based models have been trained for. The event-based model performs much better due to the minor domain-shift on the input representation.

5.3.5 Conclusions

This work includes the first results on semantic segmentation using event camera information. We build an encoder-decoder architecture which is able to learn semantic segmentation only from event camera data. Since there is no benchmark available for this problem, we propose how to generate automatic but approximate semantic segmentation labels for some sequences of the DDD17 event-based dataset. Our evaluation shows how this approach allows the effective learning of semantic segmentation models from event data. In order to feed the model, we also propose a novel event camera data representation, which encodes both the event histogram and their temporal distribution. Our semantic segmentation experiments, comparing different representations, show that our approach allows the effective learning of semantic segmentation models and that our approach outperforms other previously used event representations, even when evaluating in different time intervals. We also compare the segmentation achieved only from event data to the segmentation from conventional images, showing their benefits, their drawbacks and the benefits of combining both sensors for this task.

Chapter 6

Conclusions

State-of-the-art methods for semantic segmentation, i.e., per-pixel image classification, usually require a lot of annotations and computational capabilities that are not always available in real-world set-ups. The goal of this thesis was to overcome some of the challenges that real-world scenarios and applications introduce for semantic segmentation, in particular the limited amount of labeled data, the computation and speed restrictions and, the availability of different sensors. This section provides a summary of the major findings of this thesis research and drafts potential paths for future developments, since there are still significant challenges ahead for automated scene understanding in real-world scenarios.

6.1 Semantic Segmentation with Limited Labeled Data

The first two chapters of this thesis have addressed the common and important scenario of having limited labeled data. Current top-performing solutions for semantic segmentation require a large set of labeled data, since most of these methods are supervised deep learning-based approaches. A frequent problem to deploy these semantic segmentation methods for real-world applications is that there is not always enough labeled data available and the annotation cost is not always affordable. As described in the introduction, in Section 1.2.1, we consider two main strategies to reduce the annotation requirements when learning semantic segmentation models. In Chapter 2, we have proposed a weakly-supervised method that learns from weak labels, i.e., sparsely labeled pixels. In Chapter 3 we have proposed a novel approach for semi-supervised scenarios. In the following subsections we further discuss and analyze the presented contributions of these chapters.

Semantic Segmentation from Few Labeled Pixels: Weakly-supervised Learning.

In the first part of the thesis, we have addressed the problem of learning semantic segmentation models from sparse labels. In Chapter 2 we have presented CoralSeg [13]. This novel approach enables effective training of semantic segmentation models from a set of a few sparsely labeled pixels per image, providing similar results to those obtained when training with dense labels. The key step in our method is to train with pseudo-labels obtained by propagating the sparse pixel annotations with a novel iterative superpixel-based method. Our iterative version (multi-level) outperforms the non-iterative previous methods by a large margin. The limitations of our proposed approach

are related to the performance versus execution time trade-off. The more iterations our algorithm does, the better performance it gets but the execution time also increases. Besides, another important limitation to take into account is that our method relies on superpixels, i.e., the images have to have clear gradients for our method to perform well. Our core experimentation was run on a realistic and challenging scenario, i.e., underwater coral reef monitoring data. This is our main set-up as it presents a challenging and real-world use case where most of the available labeling efforts (made by experts) consist of sparse labels. For this use case, we have also released a generic encoder for coral imagery, trained on over half a million coral reef images. In our experiments, we show that this coral-generic encoder can be used to improve the performance of semantic segmentation models for coral reef regions with less data, by using this model as pre-training. This is similar to the wide-spread [56] pre-training strategy for current general purpose classification, detection and segmentation methods. Apart from this coral reef scenario, we have also shown the effectiveness of our approach for other robotic applications, in order to show the applicability to other domains.

We have shown the powerful aid that pseudo-labels have in deep-learning based models, this is something that has been explored a lot lately not only in semantic segmentation but in other computer vision tasks like detection or classification [39, 183, 220]. These techniques are especially effective when there is a lack of labeled data or, a lot of unlabeled data available. We have also worked on releasing to the community a pre-trained model that could serve as the ImageNet pre-trained encoders for coral detection, segmentation, classification or other visual task.

Semantic Segmentation from Partially Labeled Dataset: Semi-supervised Learning. This part of the thesis has tackled the semantic segmentation problem in a semi-supervised scenario which is a very common scenario for real-world set-ups. Semi-supervised learning aims to learn a model assuming only a small subset of the available data is labeled, tackling the limited labeled data issue by extracting knowledge from unlabeled samples. In [12] we present a novel approach based on a novel representation learning module. This module, based on contrastive learning, enforces the segmentation network to yield similar pixel-level feature representations for same-class samples across the whole dataset. To achieve this, we maintain a memory bank continuously updated with feature vectors from labeled data. These features are selected based on their quality and relevance for contrastive learning. In an end-to-end training, the features from both labeled and unlabeled data are optimized to be similar to same-class samples from the memory bank. We demonstrate that our approach outperforms the current state-of-the-art for semi-supervised semantic segmentation and semi-supervised domain adaptation on well-known public benchmarks, with larger improvements on the most challenging scenarios, i.e., less available labeled data.

Our method reduces the computation by having an small memory bank. Therefore, without any computation or memory restrictions, our method could potentially produce even better results. The proposed per-pixel contrastive learning idea can be applied to other segmentation tasks, like instance or panoptic segmentation, or even to the object detection problem. Contrastive learning [87] started to become popular for self-supervised representation learning [43, 83, 259, 268], but this last year, several papers have shown the benefits and potential of contrastive learning for more complex tasks [241, 249]. Contrastive learning is still a growing field and a research topic with still

huge potential.

6.2 Efficient Semantic Segmentation

Another very important challenge we tackled in this thesis is computational efficiency, which is a key to deploy semantic segmentation models in different real-world set-ups. In Chapter 4, we have focused on this very relevant topic, in order to make semantic segmentation affordable and accessible for more real-world applications. Many applications, such as medical robotics, drone delivery, mobile applications, just to name a few, have efficiency restrictions. The most common restrictions are related to the available computation and speed (frame-rate) requirements. Current state-of-the-art for semantic segmentation is based on large deep learning models that require high-end GPUs to be able to run at acceptable frame-rates. In this part of the thesis, we have tackled the topic of efficient semantic segmentation. We have proposed three novel architectures for efficient semantic segmentation, MiniNet [10] and *MiniNet-v2-cpu* for operating in the CPU and, *MiniNet-v2* [11] for operating in low-end GPUs. The design of the architectures we proposed is based on a study of the most relevant techniques for building efficient convolutional neural networks. The MiniNet-v2 versions make use of the proposed multi-dilation depthwise separable convolution which is one of the main contributions of this part of the thesis. We have evaluated the architectures with well-known and standard benchmarks showing the benefits they provide and making more available the use of semantic segmentation to real-world applications. To demonstrate this we have applied our architectures in a real use case: keyframe selection for VSLAM systems. For this, we have proposed a novel keyframe selection that can be integrated with state-of-the-art VSLAM systems to boost the usefulness of the keyframes. A key ingredient of the keyframe selection is the proposed MiniNet architecture, which analyzes the frames online at the robot onboard CPU. Evaluating the shared keyframes in the GPU-enabled server, we get better performance on additional tasks, text recognition in our experiments, while we do not lose the capacity of recognizing revisited places using those keyframes, essential for VSLAM systems.

Although the proposed models are more memory, computation and speed efficient than previous methods, they still perform worse than the top-performing large methods. In summary, we have shown different strategies to reduce the computation load while decreasing very little the segmentation performance. These strategies can be applied to any top-performing segmentation model allowing to reduce their computation and memory requirements and boosting their frame-rates, making segmentation models more accessible for real-world applications.

6.3 Semantic Segmentation with Other Sensors

Plenty of real-world applications run on hardware platforms equipped with various sensor types and modalities. In the last part of this thesis we have explored the possibilities and challenges for semantic segmentation solutions when using other types of sensors than RGB cameras, namely LiDAR and event-based cameras.

3D sensor: LiDAR Semantic Segmentation. LiDAR is a sensor that provides a sparse 3D representation of the scene surrounding the sensor. It is a very common sen-

sensor in robotic platforms and it is becoming so popular that even the new iPad (2020)¹ has one LiDAR integrated. Common 3D semantic segmentation approaches tend to be too slow for many robotics or interactive applications. This is because 3D information tends to be more expensive to manage, mainly because of the size of the 3D data structure and its order-less nature. Tackling this issue, we have proposed 3D-MiniNet [9], a novel and efficient approach for LiDAR semantic segmentation. The main idea of this method is first to learn a 2D representation from the 3D unordered point cloud, using 3D learnable operations, and then to make use of an efficient 2D segmentation network. In this work, we make use of MiniNet-v2 as the 2D CNN. The last step of our approach is to take the 2D output segmentation and to re-project back to the 3D point cloud. 3D-MiniNet has been evaluated and validated on well-known public benchmarks, where it gets state-of-the-art results while being faster and more parameter-efficient than previous methods.

Although 3D-MiniNet get high performances compared to previous works, there is a trade-off between the speed and performance. Ideally, the use of 3D operations for 3D data is preferable and the use of the 2D learned representation and 2D convolutions is mainly for making the network run at acceptable frame-rates.

3D sensor: LiDAR Domain Adaptation. In the second part of Chapter 5 we have proposed how to perform domain adaptation of LiDAR semantic segmentation. A common issue of machine learning systems is that they tend to overfit on the training data and have a performance drop on out-of-distribution data. For 3D data, in the case of LiDAR, the two main causes of this come from the captured data and the capturing system. Addressing this problem, we have proposed simple but effective strategies to reduce this effect by aligning the data distribution on the input space. Besides, we have also presented a learning-based module to align the distribution of the semantic classes of the target domain to the source domain.

Nevertheless, the proposed class-alignment distribution introduces an strong assumption (the source and target domain have similar class distributions) that is not always true and, therefore, cannot always be applied.

In brief, in this chapter, we have shown a novel approach for LiDAR semantic segmentation that can run in real-time and get top-performing results. We also have shown how to reduce the domain shift problem with this type of data and, we have released the code to help the research community.

Event-based cameras. The last part of Chapter 5 have tackled the semantic segmentation task for event-based cameras, where the main challenge has been the lack of labeled data due to the fact that some type of sensor data is not easy to annotate. Event-based cameras are promising sensors that register intensity changes in the captured environment. In contrast to conventional cameras, this sensor does not acquire images at a fixed frame-rate. These cameras, as their name suggests, capture events and record a stream of asynchronous events. This sensor has several advantages over the RGB camera, but, the easiness of annotation is not one of them. This type of camera is hard for a human to understand and recognize, especially at a pixel-level, what is in the image. We have proposed to use a grayscale camera synchronized to the event-camera to tackle this lack of annotations. Besides, instead of annotating the grayscale

¹<https://www.apple.com/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackp>

camera, we even reduce to zero the annotation cost by training a segmentation model on RGB data and getting pseudo-labels with this trained model [8]. We have evaluated different types of event representations for the semantic segmentation task. We have also evaluated the performance gap between grayscale cameras and event data, and we have shown that they contain complementary information performing better when they are joined together.

Event-based cameras have still less visual information than grayscale or RGB cameras, especially when there is no scene movement since they record changes in the illumination and scene. Therefore, a static scene yields zero events. Therefore, event data cannot be treated the same as RGB or grayscale information requiring some memory to keep past information. We have made the code and data available since we have presented the first semantic segmentation work with event-based cameras, showing that event-based cameras can be used for semantic segmentation.

6.4 Overall contribution.

We can draw a general conclusion for this thesis by writing that we tackle several problems that hinder the applicability of semantic segmentation to real-world applications such as lack of labels, computation restrictions and, domain adaptation. We have proposed several approaches that address these problems and that facilitate the application of semantic segmentation to a wider range of applications and scenarios.

We have made available all the code for replicating our methods and experiments in order to help the research community. Apart from making the code available we also open-sourced the trained models and used datasets. Our works and code have started to show some impact to the community. Nevertheless, the tackled problems are not completely solved yet and there is still room for improvement for future work.

6.5 Future Work

Deep learning based techniques have already brought a great advance for the semantic segmentation problem, and for solutions towards scene understanding in general. Nevertheless, the current research scenario in the community suggests that deep learning methods can still provide better performances and open new horizons and applications.

The future of semantic segmentation is getting closer and closer with instance segmentation, being joined into panoptic segmentation [124]. Common panoptic segmentation methods join object detection techniques and models with segmentation approaches [124,264] ending up with a slow and big deep learning model. New directions are heading to end-to-end approaches tackling the problem as a joined task [248]. This last work make use of attention mechanisms like transformers [242] which is currently a hot topic. These novel approaches like visual transformers [138] or MLP-based [236] architectures are showing that there are deep learning architectures different from convolutional networks that can get state of the art performances in visual tasks, including semantic segmentation [114].

Focusing on the topics that we tackled in this thesis, i.e., the limitations and challenges that real-world scenarios introduce to solve semantic segmentation tasks, there is also still a lot of open paths for research. For example, a lot of works have tackled

the semantic segmentation problem from weak supervision. A very recent example is Zhang et al. [276] which gets very impressive results on object segmentation with just the input of the surrounding bounding box and one pixel on the center of the object. Nevertheless, these works, including our work, focus on RGB data. Very few works tackle the labeling propagation or learning from weak labels in 3D data which is a more difficult and challenging task. Besides, as we have shown in our semi-supervised semantic segmentation work, contrastive learning has huge potential when learning from none or few labels. Contrastive learning has been barely exploit on this kind of tasks and it can provide real improvements. Novel and future contrastive learning techniques and pre-training techniques can really boost tasks like semantic segmentation from few labels too. A very recent work [241] has shown for the first time thanks to these techniques that semantic segmentation could be perform with no labels.

Regarding efficiency, the 2021 current trend are transformers-based architectures [242]. These type of architectures come from the Natural Language Processing field where they pushed the state-of-the-art. In computer vision, they started to perform at the same level than CNNs during this past year. These type of architectures are performing pretty well as a backbone or encoder for vision tasks although they are still a little behind CNNs for the time being. In the future, these architecture could potentially boost he performance of vision tasks and maybe, open a sub-field of efficient networks more efficient than CNNs. Regarding efficient CNNs, the very recent work [139] shows that a very efficient solution that performs pretty well is having data-dependent convolutional kernels, i.e., the applied kernel will depend on the input data in stead of being a fixed learned kernel. For efficiency, one thing to take into account is the hardware since the inference time of the operation will depend on how optimized are this operation in the hardware. Therefore, new operations need the support of hardware manufactures to also research on how to optimize them. Regarding LiDAR semantic segmentation, lot of recent works have followed the idea of combining 3D operations on the raw point cloud with 2D convolutions [77, 126], especially for efficient LiDAR segmentation, while state-of-the-art is currently obtained with only 3D operations [292]. For LiDAR data, the more recent works and the highest potential research branch is learning or proposing novel representations and operations for this type of data. 3D data is still an unsolved field where there is non a standard or best learning operation or even a representation.

References

- [1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.
- [2] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4981–4990, 2018.
- [3] Derya Akkaynak and Tali Treibitz. A revised underwater image formation model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6723–6732, 2018.
- [4] Derya Akkaynak and Tali Treibitz. Sea-thru: A method for removing water from underwater images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1682–1691, 2019.
- [5] Iñigo Alonso, Ana Cambra, Adolfo Munoz, Tali Treibitz, and Ana C Murillo. Coral-segmentation: Training dense labeling models with sparse ground truth. In *IEEE International Conference on Computer Vision Workshops*, pages 2874–2882, 2017.
- [6] Iñigo Alonso, Luis Riazuelo Montesano, Ana C Murillo, et al. Domain adaptation in lidar semantic segmentation. *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2021.
- [7] Iñigo Alonso and Ana C Murillo. Semantic segmentation from sparse labeling using multi-level superpixels. In *IEEE International Conference on Intelligent Robots and Systems*, 2018.
- [8] Iñigo Alonso and Ana C Murillo. Ev-segnet: Semantic segmentation for event-based cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- [9] Iñigo Alonso, Luis Riazuelo, Luis Montesano, and Ana C Murillo. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. *IEEE Robotics and Automation Society, RAL*, 2020.
- [10] Iñigo Alonso, Luis Riazuelo, and Ana C Murillo. Enhancing v-slam keyframe selection with an efficient convnet for semantic analysis. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4717–4723. IEEE, 2019.
- [11] Iñigo Alonso, Luis Riazuelo, and Ana C Murillo. Mininet: An efficient semantic segmentation convnet for real-time robotic applications. *IEEE Transactions on Robotics (T-RO)*, 2020.

- [12] Iñigo Alonso, Alberto Sabater, David Ferstl, Luis Montesano, and Ana C Murillo. Semi-supervised semantic segmentation with pixel-level contrastive learning from a class-wise memory bank. *arXiv preprint arXiv:2104.13415*, 2021.
- [13] Iñigo Alonso, Matan Yuval, Gal Eyal, Tali Treibitz, and Ana C Murillo. Coralseg: Learning coral segmentation from sparse annotations. *Journal of Field Robotics*, 36(8):1456–1477, 2019.
- [14] Jose Alvarez and Lars Petersson. DecomposeMe: Simplifying convnets for end-to-end learning. *arXiv preprint arXiv:1606.05426*, 2016.
- [15] Kenneth R.N. Anthony. Coral reefs under climate change and ocean acidification: Challenges and opportunities for management and policy. *Annual Review of Environment and Resources*, 41(1):59–81, 2016.
- [16] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.
- [17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2017.
- [18] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What’s the point: Semantic segmentation with point supervision. In *European Conference on Computer Vision*, pages 549–565. Springer, 2016.
- [19] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [20] Oscar Beijbom, Peter J Edmunds, David I Kline, B Greg Mitchell, and David Kriegman. Automated annotation of coral reef survey images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [21] Oscar Beijbom, Tali Treibitz, David I Kline, Gal Eyal, Adi Khen, Benjamin Neal, Yossi Loya, B Greg Mitchell, and David Kriegman. Improving automated annotation of benthic survey images using wide-band fluorescence. *Scientific Reports*, 6, 2016.
- [22] Dana Berman, Tali Treibitz, and Shai Avidan. Diving into hazelines: Color restoration of underwater images. In *Proceedings of the British Machine Vision Conference*, volume 1, 2017.
- [23] Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [24] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019.
- [25] Pierre Biasutti, Vincent Lepetit, Jean-Francois Aujol, Mathieu Brédif, and Aurélie Bugeau. LU-Net: An efficient network for 3d lidar point cloud semantic segmentation based on end-to-end-learned 3d features and u-net. In *Proceedings*

- of the *IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.
- [26] Pierre Biasutti, Vincent Lepetit, Jean-Francois Aujol, Mathieu Brédif, and Aurélie Bugeau. Lu-net: An efficient network for 3d lidar point cloud semantic segmentation based on end-to-end-learned 3d features and u-net. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.
- [27] Jonathan Binas, Daniel Neil, Shih-Chii Liu, and Tobi Delbruck. Ddd17: End-to-end davis driving dataset. *ICML Workshop on Machine Learning for Autonomous Vehicles*, 2017.
- [28] Alexandre Briot, Prashanth Viswanath, and Senthil Yogamani. Analysis of efficient cnn design techniques for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 663–672, 2018.
- [29] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.
- [30] DEP Bryant, A Rodriguez-Ramirez, S Phinn, M González-Rivero, KT Brown, BP Neal, O Hoegh-Guldberg, and S Dove. Comparison of two photographic methodologies for collecting and analyzing the condition of coral reef ecosystems. *Ecosphere*, 8(10):e01971, 2017.
- [31] Yuanzhouhan Cao, Chunhua Shen, and Heng Tao Shen. Exploiting depth from single monocular images for object detection and semantic segmentation. *IEEE Transactions on Image Processing*, 26(2):836–846, 2017.
- [32] Herman SJ Cesar. Coral reefs: their functions, threats and economic value. *Collected Essays on the Economics of Coral Reefs*, pages 14–39, 2000.
- [33] Liyi Chen, Weiwei Wu, Chenchen Fu, Xiao Han, and Yuntao Zhang. Weakly supervised semantic segmentation with boundary exploration. In *European Conference on Computer Vision*, pages 347–362. Springer, 2020.
- [34] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.
- [35] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [36] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv:1802.02611*, 2018.
- [37] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint:1802.02611*, 2018.
- [38] Minghao Chen, Hongyang Xue, and Deng Cai. Domain adaptation for semantic segmentation with maximum squares loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2090–2099, 2019.

- [39] Nicholas FY Chen. Pseudo-labels for supervised learning on dynamic vision sensor data, applied to object detection under ego-motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 644–653, 2018.
- [40] Ping-Yu Chen, Chi-Chung Chen, LanFen Chu, and Bruce McCarl. Evaluating the economic damage of climate change on global coral reefs. *Global Environmental Change*, 30:12–20, 2015.
- [41] Steven W Chen, Guilherme V Nardari, Elijah S Lee, Chao Qu, Xu Liu, Roseli Ap Francelin Romero, and Vijay Kumar. Sloam: Semantic lidar odometry and mapping for forest inventory. *IEEE Robotics and Automation Letters*, 5(2):612–619, 2020.
- [42] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [43] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.
- [44] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.
- [45] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, 2016.
- [46] François Chollet. Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1800–1807, 2017.
- [47] Patrick Ferdinand Christ, Mohamed Ezzeldin A Elshaer, Florian Ettliger, Sunil Tataavarty, Marc Bickel, Patrick Bilic, Markus Rempfler, Marco Armbruster, Felix Hofmann, Melvin D’Anastasi, et al. Automatic liver and lesion segmentation in ct using cascaded fully convolutional neural networks and 3d conditional random fields. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 415–423. Springer, 2016.
- [48] Christian, M.Mertz, and R.Mester. Contour-relaxed superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 280–293, 2013.
- [49] Joseph H Connell, Terence P Hughes, Carden C Wallace, Jason E Tanner, Kyle E Harms, and Alexander M Kerr. A long-term study of competition and diversity of corals. *Ecological Monographs*, 74(2):179–210, 2004.
- [50] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of IEEE Conference on ComputerVision and Pattern Recognition*, pages 3213–3223, 2016.
- [51] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 2015.
- [52] Jifeng Dai, Kaiming He, and Jian Sun. Boxesup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international Conference on Computer Vision*, pages 1635–1643, 2015.

- [53] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.
- [54] Luis Fernando de Souza Cardoso, Flávia Cristina Martins Queiroz Mariano, and Ezequiel Roberto Zorzal. A survey of industrial augmented reality. *Computers & Industrial Engineering*, 139:106159, 2020.
- [55] Clément Dechesne, Clément Mallet, Arnaud Le Bris, and Valérie Guet-Brunet. Semantic segmentation of forest stands of pure species combining airborne lidar data and very high resolution multispectral imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 126:129–145, 2017.
- [56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.
- [57] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with dropout. *arXiv preprint arXiv:1708.04552*, 2017.
- [58] Ayush Dewan and Wolfram Burgard. Deeptemporalseg: Temporally consistent semantic segmentation of 3d lidar scans. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2624–2630. IEEE, 2020.
- [59] Thibaut Durand, Taylor Mordan, Nicolas Thome, and Matthieu Cord. Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [60] Clinton B Edwards, Yoan Eynaud, Gareth J Williams, Nicole E Pedersen, Brian J Zgliczynski, Arthur CR Gleason, Jennifer E Smith, and Stuart A Sandin. Large-area imaging reveals biologically driven non-random spatial patterns of corals at a remote reef. *Coral Reefs*, 36(4):1291–1305, 2017.
- [61] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [62] Gal Eyal, Jörg Wiedenmann, Mila Grinblat, Cecilia D’Angelo, Esti Kramarsky-Winter, Tali Treibitz, Or Ben-Zvi, Yonathan Shaked, Tyler B Smith, Saki Harii, et al. Spectral diversity and regulation of coral fluorescence in a mesophotic reef habitat in the Red Sea. *PLoS one*, 10(6):e0128697, 2015.
- [63] Katharina E Fabricius. Effects of terrestrial runoff on the ecology of corals and coral reefs: review and synthesis. *Marine Pollution Bulletin*, 50(2):125–146, 2005.
- [64] Zhengyang Feng, Qianyu Zhou, Guangliang Cheng, Xin Tan, Jianping Shi, and Lizhuang Ma. Semi-supervised semantic segmentation via dynamic self-training and class-balanced curriculum. *arXiv preprint arXiv:2004.08514*, 2020.
- [65] Zhengyang Feng, Qianyu Zhou, Qiqi Gu, Xin Tan, Guangliang Cheng, Xuequan Lu, Jianping Shi, and Lizhuang Ma. Dmt: Dynamic mutual training for semi-supervised learning. *arXiv preprint arXiv:2004.08514*, 2020.
- [66] S Finney and J Stephen. Photo mosaics in shallow water environments: Challenges and results. *WIT Transactions on the Built Environment*, 79, 2005.

- [67] Geoff French, Samuli Laine, Timo Aila, and Michal Mackiewicz. Semi-supervised semantic segmentation needs strong, varied perturbations. In *29th British Machine Vision Conference, BMVC 2020*, 2019.
- [68] King-Sun Fu and JK Mui. A survey on image segmentation. *Pattern recognition*, 13(1):3–16, 1981.
- [69] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019.
- [70] Guillermo Gallego, Jon EA Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based, 6-dof camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2402–2412, 2018.
- [71] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *IEEE Int. Conference Comput. Vis. Pattern Recog.(CVPR)*, volume 1, 2018.
- [72] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.
- [73] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.
- [74] Weifeng Ge, Sheng Guo, Weilin Huang, and Matthew R Scott. Label-penet: Sequential label propagation and enhancement networks for weakly supervised instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3345–3354, 2019.
- [75] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.
- [76] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *International Conference on Learning Representations*, 2019.
- [77] Martin Gerdzhev, Ryan Razani, Ehsan Taghavi, and Bingbing Liu. Tornado-net: multiview total variation semantic segmentation with diamond inception module. *arXiv preprint arXiv:2008.10544*, 2020.
- [78] Jose Nuno Gomes-Pereira, Vincent Auger, Kolja Beisiegel, Robert Benjamin, Melanie Bergmann, David Bowden, Pal Buhl-Mortensen, Fabio C De Leo, Gisela Dionísio, Jennifer M Durden, et al. Current and future trends in marine image annotation software. *Progress in Oceanography*, 149:106–120, 2016.
- [79] Manuel González-Rivero, Pim Bongaerts, Oscar Beijbom, Oscar Pizarro, Ariell Friedman, Alberto Rodríguez-Ramírez, Ben Upcroft, Dan Laffoley, David Kline, Christophe Bailhache, et al. The Catlin Seaview survey–kilometre-scale seascape assessment, and monitoring of coral reef ecosystems. *Aquatic Conservation: Marine and Freshwater Ecosystems*, 24(S2):184–198, 2014.

- [80] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.
- [81] Thomas F Goreau. The ecology of Jamaican coral reefs i. species composition and zonation. *Ecology*, 40(1):67–90, 1959.
- [82] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17:529–536, 2004.
- [83] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.
- [84] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.
- [85] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [86] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.
- [87] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.
- [88] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [89] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.
- [90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.
- [91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [92] Sebastian J Hennige, David J Smith, Sarah-Jane Walsh, Michael P McGinley, Mark E Warner, and David J Suggett. Acclimation and adaptation of scleractinian coral communities along environmental gradients within an Indonesian reef system. *Journal of Experimental Marine Biology and Ecology*, 391(1-2):143–152, 2010.
- [93] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

- [94] Jarrod C Hodgson, Shane M Baylis, Rowan Mott, Ashley Herrod, and Rohan H Clarke. Precision wildlife monitoring using unmanned aerial vehicles. *Scientific Reports*, 6:22574, 2016.
- [95] Ove Hoegh-Guldberg, Peter J Mumby, Anthony J Hooten, Robert S Steneck, Paul Greenfield, Edgardo Gomez, C Drew Harvell, Peter F Sale, Alasdair J Edwards, Ken Caldeira, et al. Coral reefs under rapid climate change and ocean acidification. *Science*, 318(5857):1737–1742, 2007.
- [96] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998, 2018.
- [97] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [98] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.
- [99] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to segment every thing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4233–4241, 2018.
- [100] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.
- [101] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [102] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: Bridging the supervised and self-supervised learning. *arXiv preprint arXiv:2101.08732*, 2021.
- [103] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.
- [104] Terry P Hughes, Andrew H Baird, David R Bellwood, Margaret Card, Sean R Connolly, Carl Folke, Richard Grosberg, Ove Hoegh-Guldberg, Jeremy BC Jackson, Janice Kleypas, et al. Climate change, human impacts, and the resilience of coral reefs. *Science*, 301(5635):929–933, 2003.
- [105] Terry P Hughes, Michele L Barnes, David R Bellwood, Joshua E Cinner, Graeme S Cumming, Jeremy BC Jackson, Joanie Kleypas, Ingrid A Van De Leemput, Janice M Lough, Tiffany H Morrison, et al. Coral reefs in the anthropocene. *Nature*, 546(7656):82, 2017.
- [106] Terry P Hughes, James T Kerry, Andrew H Baird, Sean R Connolly, Andreas Dietzel, C Mark Eakin, Scott F Heron, Andrew S Hoey, Mia O Hoogenboom, Gang Liu, et al. Global warming transforms coral reef assemblages. *Nature*, 556(7702):492, 2018.

- [107] Wei-Chih Hung, Yi-Hsuan Tsai, Yan-Ting Liou, Yen-Yu Lin, and Ming-Hsuan Yang. Adversarial learning for semi-supervised semantic segmentation. *arXiv preprint arXiv:1802.07934*, 2018.
- [108] MA Huston. Patterns of species diversity on coral reefs. *Annual Review of Ecology and Systematics*, 16(1):149–177, 1985.
- [109] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- [110] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [111] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez. xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12605–12614, 2020.
- [112] Simon Jégou, Michal Drozdal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *CVPR Workshops*. IEEE, 2017.
- [113] Rui Jian, Weihua Su, Ruihao Li, Shiyue Zhang, Jiacheng Wei, Boyang Li, and Ruqiang Huang. A semantic segmentation based lidar slam system towards dynamic environments. In *International Conference on Intelligent Robotics and Applications*, pages 582–590. Springer, 2019.
- [114] Youngsaeng Jin, David Han, and Hanseok Ko. Trseg: transformer for semantic segmentation. *Pattern Recognition Letters*, 2021.
- [115] Longlong Jing, Yucheng Chen, and Yingli Tian. Coarse-to-fine semantic segmentation from image-level labels. *IEEE Transactions on Image Processing*, 29:225–236, 2019.
- [116] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.
- [117] Tarun Kalluri, Girish Varma, Manmohan Chandraker, and CV Jawahar. Universal semi-supervised semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5259–5270, 2019.
- [118] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. Cnn-based segmentation of medical imaging data. *arXiv preprint arXiv:1701.03056*, 2017.
- [119] Ronald Kemker, Carl Salvaggio, and Christopher Kanan. Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018.
- [120] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and pattern recognition*, pages 876–885, 2017.
- [121] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

- [122] Andrew King, Suchendra M Bhandarkar, and Brian M Hopkinson. A comparison of deep learning methods for semantic segmentation of coral reef survey images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1394–1402, 2018.
- [123] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [124] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019.
- [125] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 972–981, 2017.
- [126] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. Kprnet: Improving projection-based lidar semantic segmentation. *arXiv preprint arXiv:2007.12668*, 2020.
- [127] Lian Pin Koh and Serge A Wich. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. *Tropical Conservation Science*, 5(2):121–132, 2012.
- [128] Kevin E Kohler and Shaun M Gill. Coral point count with excel extensions: A visual basic program for the determination of coral and substrate coverage using random point count methodology. *Computers & Geosciences*, 32(9):1259–1269, 2006.
- [129] Alexander Kolesnikov and Christoph H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.
- [130] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.
- [131] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.
- [132] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2017.
- [133] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.
- [134] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.
- [135] JH Laxton and WJ Stablum. Sample design for quantitative estimation of sedentary organisms of coral reefs. *Biological Journal of the Linnean Society*, 6(1):1–18, 1974.
- [136] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

- [137] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.
- [138] Chen Li, Xintong Li, Xiaoyan Li, Md Mamunur Rahaman, Xiaoqi Li, Jian Wu, Yudong Yao, and Marcin Grzegorzec. A state-of-the-art survey of artificial neural networks for whole-slide image analysis: from popular convolutional neural networks to potential visual transformers. *arXiv preprint arXiv:2104.06243*, 2021.
- [139] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. *arXiv preprint arXiv:2103.06255*, 2021.
- [140] Xuyou Li, Shitong Du, Guangchun Li, and Haoyu Li. Integrate point-cloud segmentation with 3d lidar scan-matching for mobile robot localization and mapping. *Sensors*, 20(1):237, 2020.
- [141] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9145–9153, 2019.
- [142] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6936–6945, 2019.
- [143] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 db $15 \mu\text{s}$ latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.
- [144] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3159–3167, 2016.
- [145] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.
- [146] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019.
- [147] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.
- [148] Min Liu and Tobi Delbruck. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. Technical report, 2018.
- [149] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [150] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

- [151] Yossef Loya. Community structure and species diversity of hermatypic corals at Eilat, Red Sea. *Marine Biology*, 13(2):100–123, 1972.
- [152] Dengsheng Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870, 2007.
- [153] Guoliang Lu, Yiqi Zhou, Xueyong Li, and Peng Yan. Unsupervised, efficient and scalable key-frame selection for automatic summarization of surveillance videos. *Multimedia Tools and Applications*, 76(5):6309–6331, 2017.
- [154] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *International Conference on Computer Vision*, page 10, 2017.
- [155] Martin Ludvigsen, Bjørn Sortland, Geir Johnsen, and Hanumant Singh. Applications of geo-referenced underwater photo mosaics in marine biology and archaeology. *Oceanography*, 20(4):140–149, 2007.
- [156] Wenfeng Luo and Meng Yang. Semi-supervised semantic segmentation via strong-weak dual-branch network. In *European Conference on Computer Vision*, pages 784–800. Springer, 2020.
- [157] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings icml*, volume 30, page 3, 2013.
- [158] Fabiola Maffra, Zetao Chen, and Margarita Chli. Viewpoint-tolerant place recognition combining 2d and 3d information for uav navigation. In *ICRA*, 2018.
- [159] Travis Manderson, Jimmy Li, Natasha Dudek, David Meger, and Gregory Dudek. Robotic coral reef health assessment using automated image analysis. *Journal of Field Robotics*, 34(1):170–187, 2017.
- [160] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427, 2018.
- [161] N Ani Brown Mary and Dejeay Dharma. Coral reef image classification employing improved LDP for feature extraction. *Journal of Visual Communication and Image Representation*, 49:225–242, 2017.
- [162] Davide Mazzini, Marco Buzzelli, Danilo Pietro Pauy, and Raimondo Schettini. A cnn architecture for efficient semantic segmentation of street scenes. In *International Conference on Consumer Electronics-Berlin*, 2018.
- [163] Davide Mazzini and Raimondo Schettini. Spatial sampling network for fast scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- [164] Geoffrey J McLachlan. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.
- [165] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Proceedings of the european Conference on Computer Vision (ECCV)*, pages 552–568, 2018.

- [166] Robert Mendel, Luis Antonio de Souza, David Rauber, João Paulo Papa, and Christoph Palm. Semi-supervised segmentation based on error-correcting supervision. In *European Conference on Computer Vision*, pages 141–157. Springer, 2020.
- [167] Branislav Mičušík and Jana Košecká. Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision*, pages 106–119, 2010.
- [168] Andres Milioto, Philipp Lottes, and Cyrill Stachniss. Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. In *2018 IEEE International Conference on Robotics and Automation*, pages 2229–2235. IEEE, 2018.
- [169] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [170] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [171] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.
- [172] Sudhanshu Mittal, Maxim Tatarchenko, and Thomas Brox. Semi-supervised semantic segmentation with high-and low-level consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [173] Fredrik Moberg and Carl Folke. Ecological goods and services of coral reef ecosystems. *Ecological Economics*, 29(2):215–233, 1999.
- [174] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv:1611.06440*, 2016.
- [175] Md Moniruzzaman, Syed Mohammed Shamsul Islam, Mohammed Bennamoun, and Paul Lavery. Deep learning on underwater marine object detection: A survey. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 150–160. Springer, 2017.
- [176] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *arXiv preprint arXiv:1711.10288*, 2017.
- [177] Peter J Morin. *Community ecology*. John Wiley & Sons, 2009.
- [178] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.
- [179] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [180] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning*, 2010.
- [181] Siddhartha Sankar Nath, Girish Mishra, Jajnayaseni Kar, Sayan Chakraborty, and Nilanjan Dey. A survey of image classification methods and techniques. In

- 2014 *International conference on control, instrumentation, communication and computational technologies (ICCICCT)*, pages 554–557. IEEE, 2014.
- [182] Anh Nguyen, Thanh-Toan Do, Darwin G Caldwell, and Nikos G Tsagarakis. Real-time pose estimation for event cameras with stacked spatial lstm networks. *arXiv preprint arXiv:1708.09011*, 2017.
- [183] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1369–1378, 2021.
- [184] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. Hfirst: a temporal approach to object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2028–2040, 2015.
- [185] Yancheng Pan, Biao Gao, Jilin Mei, Sibogeng, Chengkun Li, and Huijing Zhao. Semanticpos: A point cloud dataset with large quantity of dynamic instances. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 687–693. IEEE, 2020.
- [186] Paul KJ Park, Baek Hwan Cho, Jin Man Park, Kyoobin Lee, Ha Young Kim, Hyo Ah Kang, Hyun Goo Lee, Jooyeon Woo, Yohan Roh, Won Jo Lee, et al. Performance improvement of deep learning based gesture recognition using spatiotemporal demosaicing technique. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 1624–1628. IEEE, 2016.
- [187] 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.
- [188] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE international Conference on Computer Vision*, pages 1796–1804, 2015.
- [189] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.
- [190] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.
- [191] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.
- [192] Xiaojuan Qi, Zhengzhe Liu, Jianping Shi, Hengshuang Zhao, and Jiaya Jia. Augmented feedback in semantic segmentation under image level supervision. In *European Conference on Computer Vision*, pages 90–105. Springer, 2016.
- [193] Sujith Ravi. Projectionnet: Learning efficient on-device deep networks using neural projections. *arXiv preprint arXiv:1708.00630*, 2017.
- [194] Marjorie L Reaka-Kudla. The global biodiversity of coral reefs: a comparison with rain forests. *Biodiversity II: Understanding and Protecting our Biological Resources*, 2:551, 1997.

- [195] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, pages 1–21, 2017.
- [196] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In *Conference on Robot Learning*, pages 969–982. PMLR, 2018.
- [197] Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017.
- [198] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [199] L. Riazuelo, J. Civera, and J. M. M. Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401–413, 2014.
- [200] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016.
- [201] Callum M Roberts, Colin J McClean, John EN Veron, Julie P Hawkins, Gerald R Allen, Don E McAllister, Cristina G Mittermeier, Frederick W Schueler, Mark Spalding, Fred Wells, et al. Marine biodiversity hotspots and conservation priorities for tropical reefs. *Science*, 295(5558):1280–1284, 2002.
- [202] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2018.
- [203] Eduardo Romera, Luis M Bergasa, Jose M Alvarez, and Mohan Trivedi. Train here, deploy there: Robust segmentation in unseen domains. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1828–1833. IEEE, 2018.
- [204] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.
- [205] 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.
- [206] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [207] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: Fast point cloud segmentation using permutohedral lattices. *arXiv preprint arXiv:1912.05905*, 2019.
- [208] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research*, 37(6):545–557, 2018.

- [209] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, pages 1163–1171, 2016.
- [210] Howard L Sanders. Marine benthic diversity: a comparative study. *The American Naturalist*, 102(925):243–282, 1968.
- [211] Carl D Schlichting, Massimo Pigliucci, et al. *Phenotypic evolution: a reaction norm perspective*. Sinauer Associates Incorporated, 1998.
- [212] P. Schmuck and M. Chli. Multi-uav collaborative monocular slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [213] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.
- [214] Tong Shen, Guosheng Lin, Chunhua Shen, and Ian Reid. Bootstrapping the performance of weakly supervised semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1363–1371, 2018.
- [215] ASM Shihavuddin, Nuno Gracias, Rafael Garcia, Arthur CR Gleason, and Brooke Gintert. Image-based coral reef classification and thematic mapping. *Remote Sensing*, 5(4):1809–1841, 2013.
- [216] Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, and Martin Jagersand. Rtseg: Real-time semantic segmentation comparative study. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1603–1607. IEEE, 2018.
- [217] Laurent Sifre and Stéphane Mallat. *Rigid-motion scattering for image classification*. PhD thesis, Citeseer, 2014.
- [218] Hanumant Singh, Jonathan Howland, and Oscar Pizarro. Advances in large-area photomosaicking underwater. *IEEE Journal of Oceanic Engineering*, 29(3):872–886, 2004.
- [219] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018.
- [220] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.
- [221] Nasim Souly, Concetto Spampinato, and Mubarak Shah. Semi supervised semantic segmentation using generative adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5688–5696, 2017.
- [222] David Ross Stoddart. Ecology and morphology of recent coral reefs. *Biological Reviews*, 44(4):433–498, 1969.
- [223] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: an evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018.
- [224] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

- [225] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. *CoRR*, abs/1607.01719, 2016.
- [226] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE International Conference on Computer Vision*, 2017.
- [227] Guolei Sun, Wenguan Wang, Jifeng Dai, and Luc Van Gool. Mining cross-image semantics for weakly supervised semantic segmentation. In *European Conference on Computer Vision*, pages 347–365. Springer, 2020.
- [228] Shuyang Sun, Liang Chen, Gregory Slabaugh, and Philip Torr. Learning to sample the most useful training patches from images. *arXiv preprint arXiv:2011.12097*, 2020.
- [229] Ying Sun, Xinchang Zhang, Qinchuan Xin, and Jianfeng Huang. Developing a multi-filter convolutional neural network for semantic segmentation using high-resolution aerial imagery and lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018.
- [230] Meng Tang, Federico Perazzi, Abdelaziz Djelouah, Ismail Ben Ayed, Christopher Schroers, and Yuri Boykov. On regularized losses for weakly-supervised cnn segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 507–522, 2018.
- [231] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.
- [232] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.
- [233] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 International Conference on 3D Vision (3DV)*, pages 537–547. IEEE, 2017.
- [234] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision*, 2010.
- [235] Peter A Todd. Morphological plasticity in scleractinian corals. *Biological Reviews*, 83(3):315–337, 2008.
- [236] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.
- [237] Stefano Trebeschi, Joost JM van Griethuysen, Doenja MJ Lambregts, Max J Lahaye, Chintan Parmar, Frans CH Bakers, Nicky HGM Peters, Regina GH Beets-Tan, and Hugo JWL Aerts. Deep learning for fully-automated localization and segmentation of rectal cancer on multiparametric mr. *Scientific reports*, 7(1):1–9, 2017.
- [238] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *IEEE International Conference on 3D Vision*, pages 11–20, 2017.

- [239] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. SEEDS: Superpixels extracted via energy-driven sampling. In *European Conference on Computer Vision*, pages 13–26, 2012.
- [240] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.
- [241] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Unsupervised semantic segmentation by contrasting object mask proposals. *arXiv preprint arXiv:2102.06191*, 2021.
- [242] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [243] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.
- [244] Aparna Nurani Venkitasubramanian, Tinne Tuytelaars, and Marie-Francine Moens. Wildlife recognition in nature documentaries with weak supervision from subtitles and external data. *Pattern Recognition Letters*, 81(C):63–70, Oct. 2016.
- [245] 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*, 2017.
- [246] Martin Visbeck. Ocean science research is key for a sustainable future. *Nature Communications*, 9(1):690, 2018.
- [247] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and pattern recognition*, pages 2517–2526, 2019.
- [248] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020.
- [249] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. Exploring cross-image pixel contrast for semantic segmentation. *arXiv preprint arXiv:2101.11939*, 2021.
- [250] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. *arXiv preprint arXiv:2011.09157*, 2020.
- [251] Yuan Wang, Tianyue Shi, Peng Yun, Lei Tai, and Ming Liu. Pointseg: Real-time semantic segmentation based on 3d lidar point cloud. *arXiv preprint arXiv:1807.06288*, 2018.
- [252] Zhonghao Wang, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-Mei Hwu, Thomas S Huang, and Honghui Shi. Alleviating semantic-level shift: A semi-supervised domain adaptation method for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 936–937, 2020.
- [253] Steven Weinberg. A comparison of coral reef survey methods. *Bijdragen tot de Dierkunde*, 51(2):199–218, 1981.

- [254] Maggie Wigness. Superlabel: A superpixel labeling interface for semantic image annotation. Technical report, Army Research Lab, Adelphi, MD, USA, 2018.
- [255] Jay M Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David Johnson, et al. Segicp: Integrated deep semantic segmentation and pose estimation. *International Conference on Intelligent Robots and Systems*, 2017.
- [256] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.
- [257] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019.
- [258] Tianyi Wu, Sheng Tang, Rui Zhang, Juan Cao, and Yongdong Zhang. Cgnet: A light-weight context guided network for semantic segmentation. *IEEE Transactions on Image Processing*, 30:1169–1179, 2020.
- [259] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.
- [260] Wei Xiang, Hongda Mao, and Vassilis Athitsos. Thundernet: A turbo unified network for real-time semantic segmentation. In *2019 IEEE Winter Applications of Computer Vision (WACV)*, pages 1789–1796. IEEE, 2019.
- [261] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.
- [262] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. A review of point cloud semantic segmentation. *arXiv preprint arXiv:1908.08854*, 2019.
- [263] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. *arXiv preprint arXiv:2011.10043*, 2020.
- [264] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.
- [265] Fisher Y. and Vladlen K. Multi-scale context aggregation by dilated convolutions. In *International Conference on learning representations*, 2016.
- [266] Yanchao Yang and Stefano Soatto. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4085–4095, 2020.
- [267] Ting Yao, Tao Mei, and Yong Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 982–990, 2016.

- [268] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE Conference on Computer Vision and pattern recognition*, pages 6210–6219, 2019.
- [269] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 325–341, 2018.
- [270] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [271] Yitian Yuan, Tao Mei, Peng Cui, and Wenwu Zhu. Video summarization by learning deep side semantic embedding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.
- [272] Sangdoon Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.
- [273] Matan Yuval, Iñigo Alonso, Gal Eyal, Dan Tchernov, Yossi Loya, Ana C Murillo, and Tali Treibitz. Repeatable semantic reef-mapping through photogrammetry and label-augmentation. *Remote Sensing*, 13(4):659, 2021.
- [274] Matan Yuval, Iñigo Alonso, Gal Eyal, Dan Tchernov, Ana C Murillo, Yossi Loya, and Tali Treibitz. Image-based mapping and semantic segmentation for depiction of coral reef community structure. In *Ocean Sciences Meeting 2020*. AGU, 2020.
- [275] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *ECCV*. Springer, 2016.
- [276] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. Interactive object segmentation with inside-outside guidance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12234–12244, 2020.
- [277] Yuhang Zhang, Richard Hartley, John Mashford, and Stewart Burn. Superpixels via pseudo-boolean optimization. In *IEEE International Conference on Computer Vision*, pages 1387–1394, 2011.
- [278] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. *arXiv preprint arXiv:2104.06490*, 2021.
- [279] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for on-line lidar point clouds semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020.
- [280] Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213. IEEE, 2019.
- [281] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Pro-*

- ceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019.
- [282] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for real-time semantic segmentation on high-resolution images. In *European Computer Vision*, 2018.
- [283] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [284] Zhihao Zhao, Wenquan Zhang, Jianfeng Gu, Junjie Yang, and Kai Huang. Lidar mapping optimization based on lightweight semantic segmentation. *IEEE Transactions on Intelligent Vehicles*, 4(3):353–362, 2019.
- [285] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. *ECCV*, 2018.
- [286] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932, 2020.
- [287] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.
- [288] Zhenjin Zhou, Lei Ma, Tengyu Fu, Ge Zhang, Mengru Yao, and Manchun Li. Change detection in coral reef environment using high-resolution images: Comparison of object-based and pixel-based paradigms. *ISPRS International Journal of Geo-Information*, 7(11):441, 2018.
- [289] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [290] Alex Zihao Zhu, Yibo Chen, and Kostas Daniilidis. Realtime time synchronized event-based stereo. *CVPR*, 2018.
- [291] Hongyuan Zhu, Fanman Meng, Jianfei Cai, and Shijian Lu. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27, 2016.
- [292] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. *arXiv preprint arXiv:2011.10033*, 2020.
- [293] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5982–5991, 2019.
- [294] Adi Zweifler, Derya Akkaynak, Tali Mass, and Tali Treibitz. In situ analysis of coral recruits using fluorescence imaging. *Frontiers in Marine Science*, 4:273, 2017.