



Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Instance Segmentation with Pixelwise Convolutional Neural Network Embeddings

Maxime Istasse

Thesis submitted in partial fulfillment
of the requirements for the degree of
Ph.D. in Engineering Sciences

Dissertation committee:

Prof. Christophe De Vleeschouwer (UCLouvain, Belgium, advisor)

Prof. Laurent Francis (UCLouvain, Belgium, chair)

Prof. Benoît Macq (UCLouvain, Belgium)

Dr. Amirafshar Moshtaghpour (Rosalind Franklin Institute, UK)

Prof. Marc Van Droogenbroeck (ULiège, Belgium)

Version of July 3, 2023.

Abstract

Instance Segmentation (IS) consists in extracting the mask of individual instances, *i.e.*, objects, present in an image. In this thesis, we address the IS problem with pixelwise embedding methods. Those methods associate embeddings, *i.e.*, descriptors, to pixels so that each instance corresponds to a specific cluster in the embedding space. In particular, we study the capability of convolutional neural networks to output appropriate embeddings in scenes where occlusions between instances result in instance partitions.

We first unify associative and spatial embedding methods in a common mathematical framework and study their strengths and limitations in challenging scenes. We then leverage this framework to design hybrid end-to-end IS methods that solve the limitations of both associative and spatial embedding methods, and still allow for robust and efficient decoding. Finally, we study the requirements of embedding-based methods to reconnect partitioned instances. We show that the frequency of partitioned cases in the training set is important, but that those can in part be synthesized through data-augmentation.

To conduct this research, our work leverages software engineering constructs, in particular multiple inheritance from object-oriented programming, to implement the different end-to-end IS methods in a common code base, in spite of the very different interdependencies between their functions. Our solution allows for their straightforward comparison and for simplified testing, visualization and maintenance.

The primitives investigated in our work are ubiquitous in other domains of computer vision, including action spotting, object tracking and pose estimation. This gives our conclusions the potential to generalize to a wide range of applications.

Acknowledgments

Like most PhD theses, mine has been a challenging journey, but the personal and technical lessons I have learned along the way are definitely worth it. I am immensely grateful for that opportunity and for the unwavering support I have received from those around me.

First and foremost, I would like to express my gratitude to my supervisor, Prof. Christophe De Vleeschouwer, for his guidance, enthusiasm, and dedication throughout these years. Many times he agreed to follow informal and elusive intuitions, the formalization and unraveling of which required patience but ended up being the most rewarding part of my work. Among his numerous qualities, Christophe has been the model of resilience and perseverance that I needed in order to complete this undertaking.

My experience as a PhD student has been greatly enriched by the collaborations with (in no particular order) Gabriel, Amir, Julien, Abolfazl, Niels, Vladimir and Prof. Kim Mens from UCLouvain; Anthony, Adrien and Prof. Marc Van Droogenbroeck from ULiège; and Davide, Marcello and Carlo from Sportradar. Thank you very much. I am also very grateful to my thesis jury for their thorough reading and invaluable feedback.

My deepest gratitude also goes to my partner, Daphné, who kept loving and believing in me despite the innumerable false *eureka* moments she witnessed; our son, Lucien, filling my heart with love and joy every single day; and our families and friends, all of whom have supported me in countless ways throughout this journey. I cannot overstate the importance of their support and benevolence in enabling me to persevere despite the changes and challenges that have occurred during these years.

University is a unique place of freedom of speech and thought. I feel very privileged to have been part of that environment, and for the discussions that have helped me shape my opinion about society, life, many of which make for very happy memories: thanks to (again, in no partic-

ular order) Pierre-Yves, Stéphanie, Beni, Anne-Sophie, Simon, Vincent, Victor, Antoine, Kévin, François, Damien, Antoine, Alexandre, Baptiste, Benoît, Gilles, Olivier, Tahani, Jean, Elliott, Prof. Laurent Jacques, Prof. Benoît Macq, Prof. Laurent Francis, Prof. Pierre Schaus, Prof. Peter Van Roy, Prof. Jean-Pierre Raskin, Prof. David Bol, and the many others I might have forgotten to include. Numerous colleagues at Keemotion and Sportradar have also been of great support: Cédric, Jacek, Damien, Alexandre, Pascaline, Valérian, Pitou, Anne-Lise, Nathan, Nathan, Luka, Antonija, as well as others I have not had the opportunity to know well yet. I am eager to get to know them better.

This thesis would not have been possible without the financial support from the UCLouvain, Keemotion, Sportradar, and from the Walloon region and its partners for the DeepSport project, nor without the help from the IT and administrative staffs of ICTEAM, ELEN, and Sportradar. I am also grateful to the CISM and CECI staffs for their fantastic support and hard work in building high-quality GPU clusters for academic research in Belgium.

Finally, this thesis could not have taken place without the online, open-source, and academic communities. I find it amazing to have had the opportunity to research such intriguing and powerful technologies and owe it, above everything else, to a context of global collaboration. As global collaboration might no longer be a given, and considering the disruptive potential of digital technologies, I am grateful to every person taking the time to utilize these thoughtfully.

Contents

Abstract	i
Acknowledgments	iii
Contents	v
List of symbols	ix
1 Introduction	1
1.1 Instance segmentation applications	4
1.2 Audience targeted by each chapter	5
1.3 Other collaborations	6
2 Background and notations	9
2.1 Instance Segmentation	9
2.1.1 Task definition	10
2.1.2 Datasets	10
2.1.3 Metrics	13
2.2 Convolutional neural networks	14
2.2.1 Neurons, layers, and networks	15
2.2.2 Translation invariance and equivariance of functions .	18
2.2.3 Training	20
2.2.4 Inference	22
2.2.5 Non-Maximum Suppression (NMS)	22
2.3 Top-down instance segmentation methods	24
2.4 Bottom-up instance segmentation methods	25
2.4.1 Associative embeddings	28
2.4.2 Spatial embeddings with Panoptic-Deeplab	31

2.4.3	Spatial embeddings with learnable centroids and instance bandwidth	34
2.5	Neural architectures supporting bottom-up instance segmentation methods	37
2.6	Conclusion	38
3	Modular output mathematics	41
3.1	Background	44
3.1.1	Network output mathematics	44
3.1.2	Multiple inheritance	46
3.2	Modular implementation of output mathematics	48
3.3	Assessment on the Panoptic-Deeplab method	53
3.3.1	Modular output mathematics	53
3.3.2	Implementing the LearnableCenters mixin	55
3.3.3	Overhead induced by the modular framework	56
3.4	Conclusion	58
4	Team segmentation using pixelwise associative embedding	61
4.1	Related works	64
4.1.1	Fully Convolutional Network (FCN)	64
4.1.2	Associative embedding	65
4.2	Team segmentation using pixelwise associative embedding . .	66
4.2.1	Team discrimination & player segmentation	66
4.3	Network architecture and supervision	67
4.3.1	Implementation details and hyperparameters	69
4.4	Experimental validation	71
4.4.1	Dataset characteristics	72
4.4.2	Evaluation metrics	72
4.4.3	Results	73
4.5	Conclusion	77
5	Associative versus spatial embeddings for instance segmentation	79
5.1	A unified mathematical framework for embedding-based methods	80
5.2	Embedding Steadiness Seed Map (ESSM)	91
5.3	Centroid-pixel push	91
5.3.1	Baselines	92
5.3.2	Direct Push-Pull Embedding Supervision (DPPES) . .	93

5.4	Centroid-Centroid Push (CCP)	96
5.5	Associative embeddings performance	96
5.6	Hybrid embeddings	100
5.6.1	Spatial embeddings with learnable centroids	101
5.6.2	Concatenated embeddings	101
5.7	Hybrid embeddings performance	104
5.7.1	Hybrid embeddings on DeepSport data	104
5.7.2	Hybrid embeddings on synthetic data	106
5.8	Broader implications	109
5.9	Conclusion	110
6	Resolving hard occlusions	113
6.1	Related works	115
6.2	Proposed Methods	116
6.2.1	A metric tailored for partitioned instances	116
6.2.2	A synthetic occlusion-controlled dataset	118
6.2.3	Ambiguity-Driven Embedding Supervision	119
6.3	Synthetic experiments	121
6.3.1	Benefit of hard occlusions during training	122
6.3.2	Benefit of more partitioned instances during training and evaluation	123
6.3.3	Benefit of color content for IS task	124
6.3.4	Benefit of shape diversity during training	125
6.4	Experiments on a real dataset	126
6.5	Conclusion	128
7	Conclusion	131
7.1	Main lessons	131
7.2	Perspectives	133
	List of publications	137
A	Appendix	139
A.1	NMS and its implementations	139
A.2	Gradient programming	140
A.3	Associative embedding convergence under centroid-pixel push- pull supervision	142
	Bibliography	147

List of symbols

x^*	Ground-truth target of prediction x
$x^\times$	Gradient truncated version of x , <i>i.e.</i> , seen as a constant by optimization process
$\mathbb{1}_{\{\text{cond}\}}$	1 if cond holds, 0 otherwise
$[x y]$	Concatenation across channels
$(x)_{\{c_1, c_2\}}$	Selection across channels
$ x _+$	Linear rectifier, <i>i.e.</i> , $\max(0, x)$
$\mathbf{X}$	Input image
H	Image height
W	Image width
P	Set of pixels positions
p	Pixel position
K	Number of semantic classes
$C(p)$	Pixelwise semantic label
$I(p)$	Pixelwise instance label
$\mathcal{I}$	Set of instance masks
$\mathcal{I}_i$	Mask of instance i
c_i	Class of instance i
$\mathcal{F}$	Foreground

LIST OF SYMBOLS

$\mathcal{B}$	Background
$\mathcal{O}$	Set of centroids
o_i	Centroid of instance i
$\tilde{o}_i^*$	Closest other centroid to o_i^*
$\Psi(p)$	Predicted pixelwise embedding
$\mathbf{H}(p)$	Predicted seed map
$\mathbf{C}(p, c)$	Predicted semantic probability
L	Global training loss
L_{emb}	Embedding loss
L_{seed}	Seed loss
L_{sem}	Semantic segmentation loss
L_{push}	Push loss
L_{pull}	Pull loss
L_{reg}	Regularization loss
λ	Loss weight
ψ	A predicted embedding
$\mathcal{P}(\psi)$	Pixel predicting embedding ψ
$\text{NMS}_{\mathcal{E}} S$	Non-Maximum Suppression, <i>i.e.</i> , the local maximas of S over the $\mathcal{E}$ domain
$2^{\mathcal{E}}$	Power set of $\mathcal{E}$, <i>i.e.</i> , the set of all subsets of $\mathcal{E}$
$\mathcal{N}$	NMS Neighborhood (for NMS)
$\mathcal{W}(p)$	Neighborhood (for Embedding Steadiness Seed Map)
t	NMS threshold
M	Supervised margin in embedding space
M_A	Supervised margin in associative embedding space

σ	Instance bandwidth
CE	Cross-entropy loss
L1	l-1 loss
L2	l-2 loss
(m)AP	(Mean) Average Precision
A1-5	Associative embeddings with 1-5 channels
C3-5	Concatenated embeddings with 3-5 channels
CCP	Centroid-Centroid Push
DPPES	Direct Push-Pull Embedding Loss
FN	False Negatives
FP	False Positives
IoU	Intersection-over-Union
LS2	Spatial embeddings with learnable centroids
PIQ	Partitioned Instance Quality
PIR	Partitioned Instance Recall
PISQ	Partitioned Instance Segmentation Quality
PP1	Push-Pull Baseline 1
PP2	Push-Pull Baseline 2
PQ	Panoptic Quality
RQ	Recognition Quality
RoI	Region-of-Interest
S2	Spatial embeddings with fixed centroids
SQ	Segmentation Quality
TP	True Positives

Chapter 1

Introduction

Instance Segmentation (IS) consists in extracting the mask of individual instances, *i.e.*, objects, present in an image. In this thesis, we study the use of pixelwise embeddings output by Convolutional Neural Networks (CNNs) to solve the IS task, with a particular focus on obtaining a robust end-to-end system capable of resolving strong occlusion cases.

The task of IS is an important primitive in computer vision systems, as it has the capability to provide, for each object in an image, an unambiguous detection to be used by other components of the system. This contrasts, in particular, with object detection, where the provided bounding boxes become ambiguous when they overlap with each other, *i.e.*, when pixels belong to different detections. Two families of methods dominate the IS literature. The first is the family of top-down methods, which first detect instances through bounding box detection and then estimate the mask for the main instance in each detection. The second family corresponds to bottom-up methods, which first associate embeddings to pixels, and cluster the embeddings to group their pixels in instance masks. As we aim to achieve maximum robustness to occlusions, which render bounding boxes ambiguous in top-down methods, we focus in this thesis on bottom-up methods.

The literature of bottom-up methods contains a lot of variety in terms of embedding spaces, loss function definitions and decoding strategies. The spatial embedding space has a structure fixed *a priori*, related to the location of each instance in the image, while the associative embedding space is based on the emergence of distinct instance descriptors through training. In both cases, ensuring that the embeddings of all instances form sufficiently distinct clusters and finding those clusters are challenging problems. Furthermore,

instance occlusions complexify the problem as they

- imply that some (discriminative) instance features become invisible;
- imply that spatially closer instances have to be assigned distinct embeddings;
- result in more borders between instances, challenging the natural smoothness of CNN predictions and the convergence of associative embeddings;
- may also result in partitioned instances, *i.e.*, instances that appear visually split in parts that have to be reconnected.

The requirements for reconnection to occur have, to our knowledge, never been studied.

Interestingly, as illustrated in Section 1.1, improvements in IS in formulation in presence of occlusions do not just benefit the IS task. The embedding space and the concept of instance in the framework of IS methods can also be twisted so as to tackle other tasks in an end-to-end fashion with minimal modifications. Being able to solve occlusions in IS has analogous implications for them.

This thesis is organized as follows. In Chapter 2, we introduce the IS task and common datasets and metrics on which to evaluate methods. We also survey neural networks and their convolutional variants, as well as the primitives necessary for decoding their outputs in IS scenarios. Representative IS methods, from which we borrow mathematical concepts and experimental setups, are presented, and network architecture requirements are discussed.

Chapter 3 proposes a novel usage of software engineering constructs to study end-to-end methods in a modular and efficient way. To this end, we regroup the set of functions computing losses, decoding the network output and computing intermediate variables under the notion of *Output Mathematics (OM)*, to be implemented as one software component of deep learning experiments. By implementing it using classes and multiple inheritance offered by object-oriented languages, we provide a modular software equivalent for the combination of partial systems of equations to define a wider mathematical system, enabling access to all variables from anywhere and automatic management of the dependencies. We demonstrate that those modular OM enable very high flexibility with respect to experiments and optimizations in the context of deep learning.

Chapter 4 extends the concept of instance to bags of objects. Specifically, it assimilates an instance to a team sport team and relies on associative embeddings to cluster basketball players into teams. We build on a CNN to learn a pixelwise descriptor that is similar for pixels belonging to players from the same team, and dissimilar when pixels correspond to distinct teams. This follows the framework of associative embeddings for IS where each team is an instance, with the particularity that it almost systematically requires the assignment of the same embedding to unconnected pixels belonging to the different players of a team. This framework enables the separation of two teams present in a single image without any per-game learning, and can be achieved using a lightweight segmentation network. Excellent results, both in terms of accuracy and generalization to new games/arenas, are achieved.

A significant part of our work compares different alternatives to define the embedding space when the density of instances increases. Previous works showed that associative embeddings struggle to perform IS when many instances are present in a scene. This has led the community to shift away from associative embeddings. In contrast, spatial embeddings build their clusters around instance locations, so that the network differentiates remote instances naturally. The counterpart is that spatial embeddings may struggle to segment instances that are close to each other.

In Chapter 5, spatial and associative embeddings are unified in a common framework, which provides a better understanding of their similarities, differences and their possible complementarity in end-to-end IS methods. We demonstrate the difficulties of training associative embeddings for IS and illustrate their failure to reach the performance level of spatial embeddings. We study hybrid embedding formulations and demonstrate that they are able to outperform spatial embedding and to lift their limitations in case of close instances.

Chapter 6 focuses on the capability of IS methods to handle occlusions between instances. In particular, it illustrates that, while occlusions are the most challenging cases in IS, common IS quality metrics do not reflect the occlusion-handling ability of methods. We therefore propose a dedicated metric specifically designed to assess the ability of methods to handle occlusions. We then synthesize a family of datasets with varying degrees of occlusion and, through a systematic set of experiments, illustrate that the ability of spatial embeddings to resolve occlusions depends on the frequency and variety of hard occlusions in the training set. The lessons drawn on synthetic data are translated into IS improvement on real data by (*i*) turning

our dataset synthesis into a dataset augmentation to increase the number of occlusions, and (ii) introducing a supervision that favors the gradients associated to struggling embeddings. The proposed tools help to better observe and improve performance on strong occlusion cases, with improved instance reconnection.

To complete this dissertation, Section 1.1 enumerates some applicative examples of the IS framework. Section 1.2 explicits the audience targeted by each chapter, as well as the related publication venues. Section 1.3 sheds some light on other collaborations that have taken place during the realization of this thesis.

1.1 Instance segmentation applications

IS is a general tool for interpreting images and, given its pixelwise level of detail, is perhaps the less ambiguous primitive to convey the presence of an object in an image to an algorithm. For this reason, it has innumerable applications across a wide number of domains that impact our everyday lives. IS can for instance be used to segment lesions or tumorous cells in medical images; atoms and bacterias in microscopy; animals from cameras mounted in the wild, and trees from satellite images so as to understand ecosystems. IS can also be utilized in logistics for maintaining inventories, estimating traffic density, or keeping track of roads, signs and obstacles in autonomous driving. IS is also a primitive of choice to select pixels of occluded people or vehicles, so as to build identity descriptors whenever possible.

In video editing, IS can be particularly useful to modify the appearance of an object, *e.g.*, to apply visual effects on it, or on its background, *e.g.*, adding statistics in sports footage. In automatic sports video production, IS can provide important features in order to estimate an adequate framing.

In addition to the direct application of the IS task, the notion of instance and embedding space in the bottom-up IS framework can be twisted so as to address other tasks, thus broadening further the range of applications. While we mostly utilize this framework to segment the spatial coverage of objects in this dissertation, events can be detected with IS across time, and objects can be tracked using spatio-temporal IS. As another example, human pose estimation can be obtained from IS and joint detection, or from IS with the pose as embedding directly. Another task to which IS transfers easily is team segmentation, tackled in Chapter 4, that enriches

the representation of the game when analyzing team sports footage.

1.2 Audience targeted by each chapter

Whilst this thesis aims at improving performance of IS methods, it discusses very different topics. It is decomposed into five principal chapters, each working on a different aspect of the IS problem. Those chapters therefore target different audiences.

The background presented in Chapter 2 is more attractive to people who are not familiar with neural networks, or with the IS task, its datasets, metrics, and methods. It tries to provide intuitions as to why CNNs constitute a convenient paradigm to tackle that task and why we prefer bottom-up methods over top-down methods for solving occlusions. It also describes representative bottom-up methods and their limitations when handling occlusions or crowded scenes.

Modular Output Mathematics (ModulOM) presented in Chapter 3 are more attractive to practitioners who are interested in implementing multiple formulations of a given task based on neural networks, or who wish to integrate interdependent tasks in a common code base in a modular fashion. The key objective is to implement the formulation of the tasks for the neural network, *i.e.*, the OM, as one modular component so as to obtain a more flexible and maintainable code base for our experiments. That chapter is, therefore, more related with software engineering and not specific to IS at all. It has been published at an ICLR 2021 workshop, suggesting that ModulOM could help more modular and incremental research on task formulations [IMV21].

Chapter 4 is mostly interesting to people willing to separate groups of similar instances, *e.g.*, teams in team sports. We demonstrate how associative embeddings can be leveraged to tackle team discrimination efficiently. Taking a step back, it illustrates that the concepts of instance and embedding space in bottom-up IS can easily be twisted so as to solve other tasks, *e.g.*, object tracking across space and time. This chapter has been published at the CVsports 2019 workshop (CVPR), proposing a robust way of making models aware of teams without game-specific fine-tuning [IMDV19].

Chapter 5 is probably most interesting to people who are familiar with the IS literature and who wish to see many bottom-up IS formulations unified in a common mathematical framework and their performances compared. We also address previous formulations' weaknesses to solve occlu-

sions, while keeping our solution as general as possible, *e.g.*, not integrating human pose or 3D scene prior. The result is a set of mathematical components that can be useful wherever the IS task can transfer. This chapter is, therefore, very oriented towards generic adjustments of the IS task formulation. This chapter should be published as a computer vision journal article seeking to unify and augment current IS bottom-up methods [IV23].

Chapter 6 illustrates that data distribution can play an important role to solve occlusions, and is perhaps most interesting to people who are willing to improve on the hardest cases of their datasets, *e.g.*, occlusions. We show that an overwhelming frequency of simple cases biases model training and evaluation metrics in such a way that a more capable task formulation may only provide marginal improvements. Hopefully, dataset augmentation provides gains that are compounding with a better task formulation on hardest cases and specific metrics can give those improvements more visibility. This chapter is expected to be published as a computer vision conference paper emphasizing the importance of data distribution and the limitations of general metrics [IMV23].

1.3 Other collaborations

Apart from the specific research on IS, the journey behind this thesis has been rhythmized by peripheral collaborations on various other topics associated to deep learning, computer vision and sports.

In [DKI⁺19], we proposed and benchmarked the Ordinal Pooling mechanism, a generalization of *max* and *average pooling* in neural networks architectures.

In [CDI⁺19], we studied how we could achieve a real-time, high-quality human segmentation thanks to a light but specific model being trained to imitate the predictions of heavier but more generic model. While the paper applied the technique to semantic segmentation, the main ideas are compatible with IS.

In [GZI⁺21], we introduced a unified model solving player segmentation, pose estimation and ball detection to analyze basketball videos. All the tasks are solved using spatial embeddings [CCZ⁺20] and part-intensity fields [KBA19]. The resulting unified method is competitive with individual models [KBA19, VZDV19, CCZ⁺20] for each task.

In [ZSI⁺22], we packaged the DeepSport dataset that we had collected and proposed several challenges: IS, ball 3D localization, camera calibration

and player re-identification.

Chapter 2

Background and notations

In this chapter, we aim to introduce the main concepts and works at the heart of the Instance Segmentation (IS) literature. We start by introducing the IS task in Section 2.1, along with datasets and metrics. Section 2.2 introduces Convolutional Neural Networks (CNNs) and related concepts that are used pervasively in the field of computer vision. Sections 2.3 and 2.4 then survey top-down and bottom-up methods that are available in the literature to solve IS. In this dissertation, we focus our investigations on bottom-up methods and, in practice, perform most of our experiments on models derived from the Panoptic-Deeplab method, introduced in-depth in Section 2.4.2. For completeness, Section 2.5 discusses the network architecture requirements necessary for competitive bottom-up IS methods. Section 2.6 concludes.

2.1 Instance Segmentation

IS, illustrated in Figure 2.1 (b), refers to the task of delineating each object in an image and optionally assigning it a semantic class [HGDG17]. It differs from semantic segmentation task [LSD15, CPK⁺17], as in Figure 2.1 (c), where objects of a semantic class are considered as a single entity. Panoptic segmentation task [KHG⁺19], as in Figure 2.1 (d), on the other hand, unifies the segmentation of *things*, *i.e.*, countable objects, *e.g.*, person, and *stuff*, *i.e.*, amorphous regions, *e.g.*, sky.

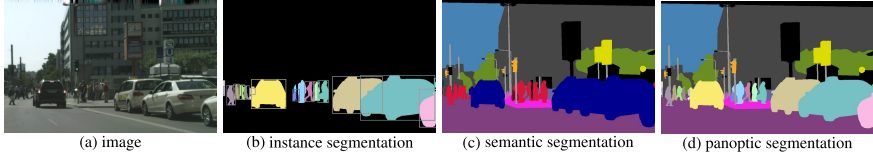


Figure 2.1: An image of the Cityscapes dataset [COR⁺16] and its different ground-truth segmentation, from [KHG⁺19]. Panoptic segmentation unifies the results of both instance and semantic segmentations.

2.1.1 Task definition

Let $\mathbf{X} \in \mathbb{R}^{H \times W \times 3} \simeq \mathbb{R}^{|P| \times 3}$ be an RGB image with pixel positions $P = [0, H) \times [0, W)$ each represented as p . The goal of IS is (i) to regroup pixels of the image into $|\mathcal{I}|$ instances masks $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \dots\}$ and (ii) to predict the semantic class of each instance i as $c_i \in \{1, \dots, K\}$, with K being the total number of classes of interest. Predicted instance masks $\mathcal{I}$ and class labels c_i should approximate as well as possible the ground-truth, *i.e.*, annotated, instances masks $\mathcal{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \dots\}$ and class labels c_j^* . While the number of classes is consistent across annotation and prediction, *i.e.*, K , there may be more or less predicted instances than annotated instances, *i.e.*, $|\mathcal{I}|$ and $|\mathcal{I}^*|$ can be different.

In the context of Panoptic Segmentation, the method must in addition predict a collection of semantic class labels for stuff pixels $C \in \{1, \dots, K\}^{|P|}$ reflecting as accurately as possible ground-truth semantic class labels $C^* \in \{1, \dots, K\}^{|P|}$.

In both cases, we regroup object pixels into the *foreground* mask $\mathcal{F}^*$, and stuff pixels into the *background*. In this dissertation, as we focus our study on the instance delineation task, it is frequent for us to work on a single-class IS setup, in which case all the instances belong to the same class, *i.e.*, *object* or *foreground*, and all the other pixels are segmented as *background*.

2.1.2 Datasets

Datasets in the state of the art of IS vary wildly in terms of number of classes, number of images, complexity of objects, scenes, and annotation quality. We introduce COCO [LMB⁺14], Cityscapes [COR⁺16] and the DeepSport [ZSI⁺22] datasets here, although many more are present in the literature [LSC12, SMFT14, EEVG⁺15, NORBK17]. COCO and

Cityscapes are the most famous ones and were often used during this research. However, their annotation quality and multi-class aspect, respectively, made it difficult for us to isolate the contribution of IS mechanisms. Other datasets that do not suffer from those problems usually showcase simplistic scenes in terms of contrast between foreground and background, or in terms of occlusions. In this context, the DeepSport dataset positions itself as a high-quality, single-class IS dataset with more complex scenes in terms of foreground, background, and occlusions.

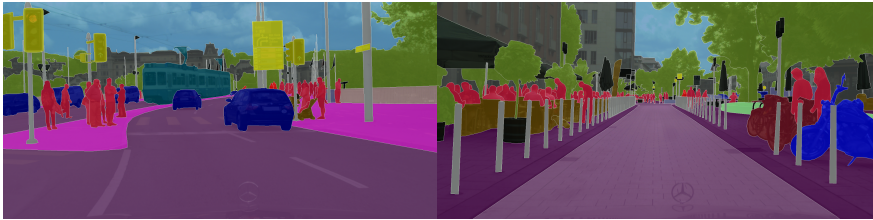
COCO. Common Objects in COntext (COCO) [LMB⁺14] is a large-scale object detection, segmentation, and image captioning dataset with images from all kinds of scenes. It is composed of 123,287 images and totalling 886 284 individually annotated instances regrouped in 80 classes. Sample images are provided in Figure 2.2 (a). While it is very varied, the annotation quality is poor as object masks are annotated with rough polygons, are lacking parts, and occlusions sometimes result in ambiguities as no depth ordering of the masks is provided.

Cityscapes. Cityscapes [COR⁺16] is a large panoptic segmentation dataset around autonomous driving. It provides a set of 5,000 finely annotated images with object masks and pixelwise semantic classes. This is thus a dataset of choice for evaluating instance and panoptic segmentation methods. It features a set of 27 classes that are very differently represented, and sometimes visually ambiguous, rendering semantic segmentation challenging by itself (person *vs.* rider, car *vs.* truck). Sample images are provided in Figure 2.2 (b).

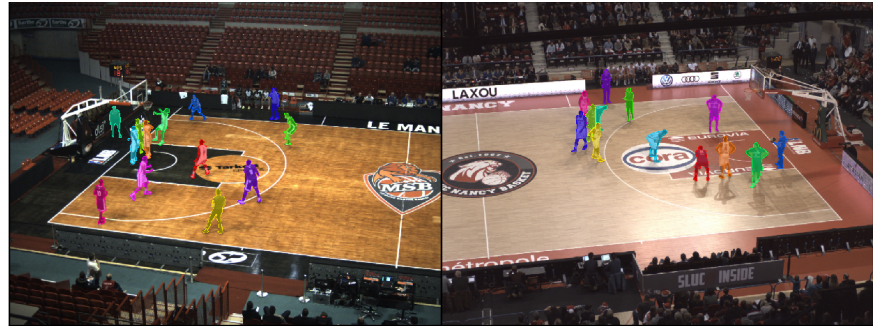
DeepSport dataset. The DeepSport basketball dataset [ZSI⁺22] was collected by our research group in the scope of the DeepSport WallInnov project. The dataset comprises around 700 images with approximately 6,000 instances, some of which are illustrated in Figure 2.2 (c). A first round of annotations was performed to annotate ball position, player skeleton and team affiliation, that we use in Chapter 4. Precise instance masks were added later and are used in Chapters 5 and 6. Similar to Cityscapes, the annotation are very precise, and the sport context allows for variety in poses, challenging occlusions and similarity in players' appearances. The single-class aspect also allows to focus on the IS problem rather than on instance classification.



(a) Examples from the COCO dataset [LMB⁺14]. COCO images are varied in terms of classes, its instance masks are annotated with rough polygons and often lack precision or parts. It is also frequent to have pixels that are assigned twice, implying ambiguity with respect to occlusions as no depth ordering is provided.



(b) Examples from the Cityscapes dataset [COR⁺16]. Cityscapes images focus on urban scenes taken from the point of view of a vehicle. Instance masks are accurately annotated and mutually exclusive. Every pixel in the image is assigned to a semantic class, which allows for Panoptic Segmentation.



(c) Examples from the DeepSport dataset [GZI⁺21]. DeepSport images focus on basketball scenes and provide high-quality IS masks, with varied player poses and occlusion patterns.

Figure 2.2: Images from COCO, Cityscapes and DeepSport datasets annotated for IS.

2.1.3 Metrics

Average Precision (AP) The average precision is a metric that originates from object detection with bounding boxes [EEVG⁺15]. In short, it corresponds to the area under the Precision-Recall curve, although slight variations exist as to how to compute it. We follow the convention in Cityscapes dataset [COR⁺16]. Computation of AP requires the ground-truth and predicted instance masks and their semantic labels as well as a set of confidence scores, *i.e.*, one confidence score per predicted instance. Let us define $AP^\tau(c)$ as the AP value for semantic class $c \in \{1, \dots, K\}$ with respect to an IoU threshold τ , which is used to perform the matching between predicted and ground-truth instances. More precisely, the matching splits the instances into three True Positive (TP), False Positive (FP), and False Negative (FN) sets. The criterion $AP^\tau := K^{-1} \sum_c AP^\tau(c)$ is then computed as the averaged AP over semantic classes. $AP^\tau(c)$ is computed as the area under the Precision-Recall (PR) curve for the class c . In the following, we explain how to compute PR curve for a given class.

Let $\mathcal{I}^*$ and $\mathcal{I}$ be the set of ground-truth and predicted instances, respectively, of a given semantic class c . Given an IoU threshold τ , the PR curve can be iteratively computed as follows¹: (i) pick the highest confidence score λ ; (ii) generate the subset $\mathcal{I}(\lambda) \subset \mathcal{I}$ containing only the predicted instances with confidence score greater than λ ; (iii) perform the matching between $\mathcal{I}^*$ and $\mathcal{I}(\lambda)$ with respect to the IoU threshold τ to identify TP set; (iv) calculate the Precision and Recall values

$$P(\lambda) = \frac{|\text{TP}(\lambda)|}{|\mathcal{I}(\lambda)|}, \quad R(\lambda) = \frac{|\text{TP}(\lambda)|}{|\mathcal{I}^*|}; \quad (2.1)$$

(v) pick the next highest confidence score and repeat (i-iv) until no confidence score is left.

Mean Average Precision (mAP). As described above, $AP^\tau(c)$ is computed with respect to an IoU threshold τ . As the TP set contains the predicted and ground-truth pairs with IoU greater than τ , AP^τ does not distinguish the cases with $\text{IoU} = \tau$ from the cases with $\text{IoU} \gg \tau$. To solve this issue, MS COCO benchmark recommends using the mean AP (mAP), *i.e.*, $mAP(c) = \frac{1}{10}(AP^{50}(c) + AP^{55}(c) + \dots + AP^{95}(c))$. It can be observed that a borderline prediction ($0.5 < \text{IoU} \leq 0.55$) is only rewarded $1/10^{th}$

¹Note that this procedure is not efficient for implementation, while it simplifies the explanation.

of a very high quality prediction. Another weakness of mAP is that it inadvertently favors methods predicting many false positives [JZD⁺23].

Panoptic Quality (PQ). Unlike AP, PQ does not require instance confidence scores but is only adapted when predicted instance masks are mutually exclusive, such as in most bottom-up IS methods. To identify the TP, FP, and FN sets, PQ computation first matches the predicted and ground-truth instances with respect to the IoU threshold $\tau = 0.5$. The PQ measure for class $c \in \{1, \dots, C\}$ can then be computed, favoring high TP instance IoUs and penalizing FP and FN instances:

$$\text{PQ}(c) = \underbrace{\frac{\sum_{(p,g) \in \text{TP}} \text{IoU}(p,g)}{|\text{TP}|}}_{=\text{SQ}(c)} \cdot \underbrace{\frac{|\text{TP}|}{|\text{TP}| + \frac{1}{2}|\text{FP}| + \frac{1}{2}|\text{FN}|}}_{=\text{RQ}(c)}. \quad (2.2)$$

As indicated, PQ can be decomposed into two components: Segmentation Quality (SQ) and Recognition Quality (RQ). Accordingly, the overall PQ, SQ, and RQ criteria can be computed by averaging the corresponding quality components over semantic classes.

Downsides of aggregating per-class metrics. For each of those metrics, the global figure is obtained by averaging class-specific metrics. Classes with a low number of instances, that have less stable metrics, therefore induce a lot of variability on the global metrics. Another downside of handling classes separately is that a classification error is penalized twice compared to a detection error: it implies a FN in the ground-truth class and a FP in the predicted class.

To allow for more direct analysis of the performance of the IS part, we therefore tend to research single class problems first.

2.2 Convolutional neural networks

In this section, we introduce fundamental concepts involved when tackling the IS task with CNNs. We first introduce neural networks and convolutional architectures in Section 2.2.1, the translation invariance and equivariance properties of functions in Section 2.2.2, that may apply or not to neural networks or functions to approximate, and what it means for CNNs. The general concepts behind the training of neural networks are presented

in Section 2.2.3, while Section 2.2.4 introduces inference, with a focus in Section 2.2.5 on the Non-Maximum Suppression (NMS) operation that is so important in the post-processing of network outputs to achieve IS.

2.2.1 Neurons, layers, and networks

Neuron or perceptron. A neuron is a mathematical unit that performs a very simple operation: given inputs $x_1, \dots, x_N$, weight parameters $w_1, \dots, w_N$, an optional bias parameter b and an optional activation function $f : \mathbb{R} \rightarrow \mathbb{R}$, it computes its *activation*

$$a = f \left(\sum_i w_i x_i + b \right) . \quad (2.3)$$

An illustration of a neuron is provided in Figure 2.3a.

The most common activation functions are illustrated in Figure 2.4. Depending on its activation function f , a neuron can be used as a binary classifier, *e.g.*, using a step, tanh, or sigmoid function that will constrain the output within some interval. An identity activation will instead allow to perform regression, *i.e.*, to approximate functions taking arbitrary values. Relying on a linear combination at their core, single neurons, however, have a very limited *expressivity*, *i.e.*, they are only able to approximate a very restricted set of functions $g : \mathbb{R}^N \rightarrow \mathbb{R}$.

Neural network or multilayer perceptron. To model more complex functions, neurons can be grouped inside networks. This is usually performed in a layered fashion, as illustrated in Figure 2.3b, so that each layer l is composed of neurons whose inputs are the outputs of neurons from the previous layer $l - 1$. We call such layers *dense*, or *fully connected* layers. The weights of the different neurons of a layer can be regrouped inside a matrix $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$, while their biases form a vector $\mathbf{B}_l \in \mathbb{R}^{N_l}$. The activations of layer l can therefore be computed using matrix multiplication

$$\mathbf{A}_l = f(\mathbf{W}_l \mathbf{A}_{l-1} + \mathbf{B}_l) , \quad (2.4)$$

where the previously presented activation functions f apply elementwise. The inputs of the first layer, $\mathbf{A}_0$, are the input signal, while the activations of the last layer, $\mathbf{A}_D$, provide the outputs. The size of the final layer and its non-linearity can be adapted to approximate arbitrary functions $f : \mathbb{R}^N \rightarrow$

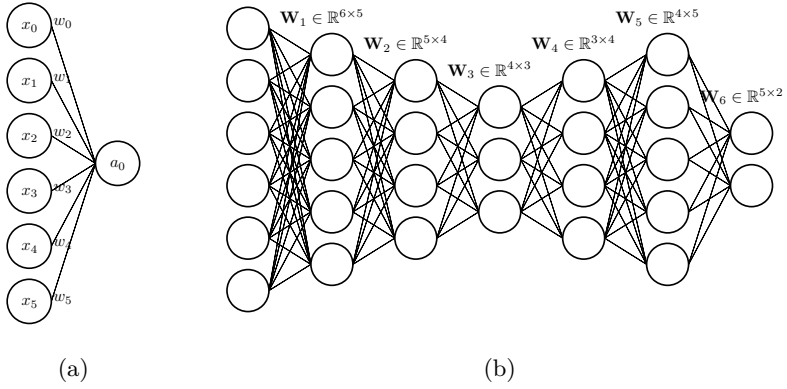


Figure 2.3: (a) Illustration of a single neuron with 6 inputs, hence 6 weights. It computes a weighted sum of the input vector and applies an activation function. This neuron can approximate functions $g : \mathbb{R}^6 \rightarrow \mathbb{R}$. (b) Illustration of a neural network made of 6 dense, fully connected layers. Each layer is an ensemble of neurons taking as input the activations of the neurons of the previous layer, except for the first layer whose neurons directly receive the input signal. The number of layers and the number of neurons in each layer are hyperparameters. This network can approximate functions $g : \mathbb{R}^6 \rightarrow \mathbb{R}^2$.

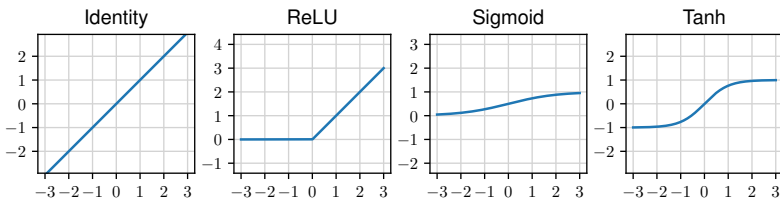


Figure 2.4: Established activation functions $f(x)$. ReLU is the most popular for inner neurons, while tanh and sigmoid functions bound the values and are most often used for the output layer. An identity activation on the output value, in contrast, allows the output of arbitrarily high values.

$\mathbb{R}^M$ with arbitrary N and M . Classification with more than two labels is usually achieved thanks to the softmax vector activation function

$$\text{Softmax}(\mathbf{A}_l)_i = \frac{e^{\mathbf{A}_{li}}}{\sum_j e^{\mathbf{A}_{lj}}} , \quad (2.5)$$

which generalizes the sigmoid activation for more than two classes.

The number of layers and the number of neurons in each inner layer will impact the expressivity of the neural network, *i.e.*, the complexity of functions that it will be able to approximate. Neural networks have been proved to be able to approximate arbitrarily complex functions arbitrarily well, provided that they are given enough neurons [HSW90, Csá01].

In spite of this universality, processing an image with a fully connected neural network presents at least two challenges. First, a high-resolution image is equivalent to a vector of very high dimension, which enforces a lot of weights to be used in the first layer. Second, every neuron in the network mixes information from the whole image, while there is usually no reason to create linear combinations of components from spatially distant pixels early. Pixels from images are best compared with their neighbors to detect local features first, *e.g.*, edges, corners, and only then can those features help to comprehend the wider content of an image.

CNNs. CNNs offer a solution to both difficulties by leveraging *convolutional layers* [LBBH98]. The easiest way to visualize a convolutional layer is as a dense layer sliding across its input image or *feature map*, as illustrated in Figure 2.5, receiving activations from a small spatial window and providing a vector output for one position every time. Once again, a CNN will stack convolutional layers to achieve more complex functions. Unless it includes fully connected layers at some point, a CNN is said to be *fully convolutional* [LSD15]. Such networks do not necessarily mix information from the whole signal. In such cases, each output neuron has a limited *receptive field*, *i.e.*, input region from which it receives information.

While a CNN can output scalars or vectors as dense neural networks, it is ideal to obtain outputs that follow the structure of the input signal, *e.g.*, 2-D maps from 2-D images, and where the neurons at a given location attend and characterize an analogous region in the input signal. A frequent example is a semantic segmentation CNN that assigns a probability to each pixel to belong to each semantic class of interest in a given application domain.

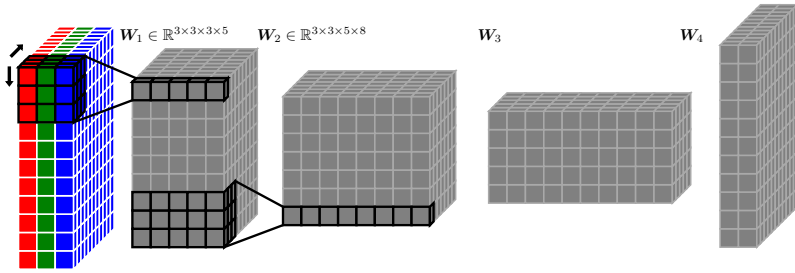


Figure 2.5: 3-D illustration of a 2-D CNN receiving a 3-channel (RGB) input and providing a 2-channel output. Each gray block is a set of feature maps, the activations of a convolutional layer, that are obtained by sliding a dense layer on its input. The number of parameters for each convolutional layer is therefore the product of the filter size, *i.e.*, 3×3 in our case, the number of input feature maps, and the number of output feature maps, *i.e.*, 3 and 5 respectively for the first layer, 5 and 8 for the second. Recovering original resolution as performed in the last layer requires an upsampling mechanism.

Architecture. The *architecture* of a neural network is its definition in terms of layers and their hyperparameters, *e.g.*, number of neurons or filters, the size of the sliding window.

Aside from dense and convolutional layers, many other layers exist, such as *Pooling* layers [BPL10, HZRS15, DKI⁺19], that help aggregating context to increase the receptive field of neurons quicker, *BatchNorm* [IS15] or *LayerNorm* [BKH16] achieving some normalization of the activations to stabilize training, *Dropout* layers that help increasing the redundancy between different neurons at training [SHK⁺14], *ConvTranspose* and *Up-Sampling* layers that allow to easily recover the input resolution for the outputs [LSD15]. Another important aspect of current architectures are *skip connections* that preserve supervision signal through deep networks and allow high-resolution information to be forwarded to high-resolution outputs [RFB15, HZRS16].

2.2.2 Translation invariance and equivariance of functions

In this section, we define translation invariance and equivariance on functions of an image $\mathbf{X} \in \mathbb{R}^{H \times W}$, and discuss their implications on CNNs.

Given a pixel position $p \in P$, we here refer to the value of $\mathbf{X}$ at that pixel as $\mathbf{X}_p$. Let us define a translation operator $T(\mathbf{X}, t)$ translating image $\mathbf{X}$ by a translation $t \in \mathbb{Z}^2$:

$$T(\mathbf{X}, t)_p = \mathbf{X}_{p+t} . \quad (2.6)$$

Translation-invariant function. A function $f : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}$ is translation invariant if its output is not affected by a translation of its input, *i.e.*,

$$f(T(\mathbf{X}, t)) = f(\mathbf{X}) \quad \forall t . \quad (2.7)$$

In such cases, there is no interest in evaluating $f(\cdot)$ with translated versions of $\mathbf{X}$, as it will lead to the same output. Translation-invariant functions are most often used to characterize images globally, *e.g.*, to classify them. Outputs in those cases are vectors with no spatial dimensions.

When the output of $f(\mathbf{X})$ has itself spatial dimensions, *e.g.*, $f(\mathbf{X}) \in \mathbb{R}^{H \times W}$, we are most often interested by having each location of the output providing information about a corresponding region in the input. In such cases, translation-equivariance may apply.

Translation-equivariant function. A function is translation-equivariant if a translation of the input results in the same output, but translated by the same amount [NALV18], *i.e.*,

$$\forall t : f(T(\mathbf{X}, t)) = T(f(\mathbf{X}), t) . \quad (2.8)$$

When outputs at different locations are generated in the same way, *e.g.*, through a local function f' evaluated on translated versions of the inputs, *i.e.*,

$$f(\mathbf{X})_p = f'(T(\mathbf{X}, p)) , \quad (2.9)$$

f is guaranteed to be translation-equivariant.

Convolutional layers, and, therefore, CNNs, work mostly in this way, *i.e.*, as functions sliding over inputs spatially, and therefore have a natural tendency to work in a translation-equivariant manner. However, implementation details and global structure in the inputs introduce deviations from translation-equivariant behavior, which can even be encouraged on purpose through supervision [LLM⁺18, BZXL19, KL20].

In any case, the advantage of CNNs working in a translation-equivariant manner is that the network can model the local function f' instead of the global f one. This may lead to a multiplication of the number of train-

ing samples, and allow less parameters to be required. In other words, CNNs approximating translation-equivariant functions are more data- and parameter-efficient.

As an example, associative embedding formulations, described in Section 2.4, *e.g.*, [NHD17, DBNVG17], require a global consensus of the pixelwise predictions across the whole image, which the CNNs cannot solve through the approximation of a local function. Spatial embedding formulations, *e.g.*, [PZC⁺18, NBPG19, CCZ⁺20], in contrast, only require CNNs to solve tasks at the scale of objects and, therefore, benefit from better data and parameter efficiency than associative embedding formulations. Chapter 5 will confirm spatial embeddings’ advantage, and propose hybrid embeddings in which the associative part can work locally, finally outperforming spatial embeddings.

It is worth mentioning that other invariance and equivariance properties exist, *e.g.*, invariance or equivariance to scale or rotation, invariance to small brightness, hue or saturation perturbations. While architectural solutions exist to help satisfying some [LAE⁺16], those properties are most often incentivized through data augmentation, quickly introduced in Section 2.2.3.

2.2.3 Training

Being initialized with random parameters, neural networks need to be trained to solve any problem. We adjust their parameters by feeding them with input samples, *i.e.*, images in our case, and evaluating a differentiable loss function L quantifying the discrepancy between the actual outputs and expected outputs, *e.g.*, outputs corresponding to annotations. Thanks to the design of neural networks, L is differentiable with respect to the network outputs $\mathbf{A}_D$, and, therefore, with respect to the network parameters $\mathbf{W}$ and $\mathbf{B}$, so that we can update them using Stochastic Gradient Descent (SGD) optimization steps:

$$\mathbf{W}_l \leftarrow \mathbf{W}_l - \mu \nabla_{\mathbf{W}_l} L \quad (2.10a)$$

$$\mathbf{B}_l \leftarrow \mathbf{B}_l - \mu \nabla_{\mathbf{B}_l} L, \quad (2.10b)$$

where $\nabla_{\mathbf{W}_l} L$ (resp., $\nabla_{\mathbf{B}_l} L$) denotes the gradient of L with respect to the weights $\mathbf{W}_l$ (resp., optional biases $\mathbf{B}_l$). The learning rate μ scales each optimization step and is a hyperparameter of the training. We now introduce terms and practices that are frequent in the literature and will be used throughout this dissertation.

Batch. Most often, the loss L is computed and averaged over multiple training samples at each optimization step. We call each of those sample sets a *batch*.

Epoch. The epoch is the unit of time or number of optimization steps that it takes for the network to be fed every sample of the dataset exactly once. It is most often used to standardize training duration.

Backpropagation. The gradient of the loss evaluated on a batch is generally computed using the backpropagation algorithm [RHW86] as it allows an efficient and exact computation of the gradient with respect to all network parameters.

Multi-loss training. Complex tasks such as IS usually involve multiple sublosses L_i in L . It is common to weigh them with constant weights λ_i set as hyperparameters during the whole training [KGC18, NBPG19, CCZ⁺20].

Learning rate schedule. Better convergence of the model parameters is often achieved by adjusting the learning-rate during training. Many different policies exist, which we refer to as *learning rate schedules*.

Data augmentation. In order to favor the emergence of desired invariance and equivariance properties in the neural network, it is frequent to randomly perturb input images before feeding them to the network. Therefore, the network sees a different version of the whole dataset at every epoch. Classical transformations include random color jitter, random cropping, scaling, flipping and small rotation perturbations [CZM⁺19].

Pre-trained backbone. Practitioners often initialize the weights of the model backbone, *i.e.*, the first few layers, with the weights of a network that has been trained on another task. This helps to shorten the training time and to obtain better performance when data is scarce [HGD19].

Online hard example mining. It is a common practice to increase the contribution of wrong or uncertain outputs to the supervision [SGG16, LGG⁺17]. The idea is that negligible errors on good predictions may otherwise acquire too much weight because of their overwhelming frequency.

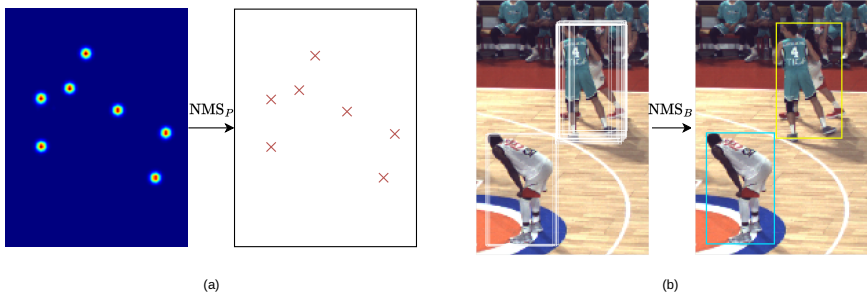


Figure 2.6: Non-Maximum Suppression illustrated on a heatmap (a) and on bounding box predictions (b). The goal of NMS is, given a set of elements $\mathcal{E}$ and scores $S : \mathcal{E} \rightarrow \mathbb{R}$, to remove elements that are redundant with similar ones that are higher-scored.

2.2.4 Inference

At inference, the image of interest is passed through the network, and its output is decoded. The exact decoding process depends on the task and on the formulation that has been chosen to train the network. Indeed, the network only sees and predicts numbers, and it is the responsibility of the practitioner to frame the problem so that the network is able to discover patterns during training, and to interpret its output accordingly at inference.

While some problems have straightforward solutions that require very little post-processing, *e.g.*, classification or semantic segmentation, IS is more complex. It is very related to the fact that the number of instances to be predicted depends on the image. IS therefore involves sparse predictions over the spatial domain, hence an *all-to-some* mapping, while only *all-to-one* and *all-to-all* mappings are trivial to implement with neural networks. Thus, an *all-to-all* mapping is used and sparsified through Non-Maximum Suppression.

2.2.5 Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is the process of removing predictions that are not local maximas given a scoring function and neighborhood definition. To our knowledge, the first use of NMS with CNNs achieved the removal of redundant predicted bounding boxes [GDDM14]. More recent works also refer to NMS when extracting local maximas from high-resolution

detection output maps [CCZ⁺20]. We illustrate both use cases in Figure 2.6. However, to our knowledge, the operation is only described in the literature as algorithms handling the specific cases of bounding boxes [BSCD17, BZXL19].

In order to take advantage from the universality of NMS in the rest of this dissertation, we introduce the NMS operator that can be seen as analogous to the argmax operator, but extracting the set of local maxima rather than the global maximum. We define the NMS operator over a set of elements $\mathcal{E}$, a scoring function $S : \mathcal{E} \rightarrow \mathbb{R}$, a threshold $t \in \mathbb{R}$ and a neighborhood $\mathcal{N} : \mathcal{E} \rightarrow 2^{\mathcal{E}}$ as follows:

$$\text{NMS}_{\mathcal{E}, \mathcal{N}, t} S = \{e \in \mathcal{E} : S(e) > t \text{ and } \forall d \in \mathcal{N}(e), S(e) \geq S(d)\} . \quad (2.11)$$

That is, each element e from $\mathcal{E}$ is in the resulting set if its score $S(e)$ is higher than the threshold, and higher than those of its neighbors.

The NMS operator being generic, the sets $\mathcal{E}$ encountered in this dissertation will be pixel locations $P \subset \mathbb{R}^2$, predicted bounding boxes $B \subset \mathbb{R}^4$ and predicted embeddings $\Psi \subset \mathbb{R}^D$, D being the embedding dimensionality. Because $\mathcal{N}$ and t are generally specific hyperparameters, we omit them from equations in favor of the more concise notations $\text{NMS}_{\mathcal{E}} S$ or $\text{NMS}_{e \in \mathcal{E}} S(e)$, and only specify them in experimental setup descriptions.

When working in the image space, *i.e.*, with pixel coordinates $p \in P$, the neighborhood $\mathcal{N}(p)$ is usually the set of positions contained in a small window centered around p . In an isotropic space of vectors $\psi \in \mathbb{R}^D$, $\mathcal{N}(\psi)$ is generally a ball of radius M centered around ψ . When working with bounding boxes coordinates $b \in \mathbb{R}^4$, the neighborhood $\mathcal{N}(b)$ is the set of predicted bounding boxes b' that have an intersection-over-union with b greater than some threshold T , typically 50% or 70%. For the sake of completeness, Appendix A.1 discusses implementation considerations that lead in some cases to negligible discrepancies from Equation (2.11) [BZXL19].

The parameters of the neighborhood always have an impact on the performance: the neighborhood should be large enough to cancel the redundant predictions corresponding to one target, without removing legitimate predictions of similar targets. This often leads to a fixed trade-off that *de facto* discards the possibility of solving crowded cases. One of the missions of Chapter 5 will be to alleviate this constraint.

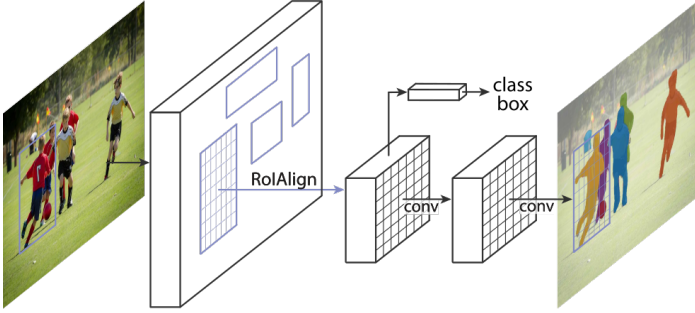


Figure 2.7: Illustration of the top-down Mask-RCNN method for IS, from [HGDG17]. Objects in the images are first detected by a bounding-box detector. Each bounding box detection is then taken as Region-of-Interest and for its main object to be segmented using the semantic segmentation paradigm. Therefore, the segmentation is impacted by ambiguous or inaccurate bounding box detections.

2.3 Top-down instance segmentation methods

The literature of IS is usually considered to be split in two: top-down methods, addressed in this section, and bottom-up methods, addressed in the next one. Top-down IS methods first detect objects of interest through a CNN-based bounding-box detector. This step consists in the prediction of a multitude of bounding box coordinates $B_i \in \mathbb{R}^4$, and a score $S(B_i)$ for each bounding box. Non-maximum suppression $\text{NMS}_B S$ is then used to keep most promising bounding boxes as detections, as illustrated in Figure 2.6 (b). Only then is each detection considered as a *Region of Interest (RoI)* for a new *neural network head*, i.e., a new sequence of layers relying on the previously built features, to segment the main object by performing semantic segmentation. Top-down IS methods are most often built on top of Mask-RCNN [HGDG17], illustrated in Figure 2.7, which itself uses Faster-RCNN [RHGS15] as detector.

Using bounding box prediction for the first phase is problematic in case of occlusions for several reasons, illustrated on Figure 2.8: (i) highly overlapping objects may see their bounding boxes cancel out each other through the NMS step, e.g., if their intersection over union is higher than the NMS threshold, and (ii) imprecise bounding boxes quickly become ambiguous as to which object they refer to. (iii) The prediction of bounding box coordinates is systematically subject to noise, which is more important when an

instance is occluded or close to other instances. Finally, using those noisy bounding boxes as RoI for the segmentation task (*iv*) introduces noise in the supervision of the segmentation as the assignment of an instance to a prediction may be unstable and (*v*) constrains the extent of the segmentation mask when the bounding box does not cover some part of the instance.

Whilst various ideas have been investigated to better handle occlusions in top-down methods [LM16, KTT21], to our knowledge, none of those fundamental weaknesses are addressed. Those issues are mitigated by striking dataset-specific trade-offs and significant hyperparameter search. Looking for methods in which most of those issues can be solved naturally, we decided to research more in the direction of bottom-up methods.

Despite those weaknesses, top-down methods have long been prevailing in public benchmarks, perhaps because of the simplistic architectures of bottom-up methods [CCZ⁺20]. Recent works however reveal that part of the advantage of top-down methods could lie in a more natural alignment with the mAP metric in the errors they favor [JZD⁺23].

2.4 Bottom-up instance segmentation methods

Bottom-up IS methods first associate an embedding vector to every pixel and then find clusters in the embedding space. The pixels whose embeddings lie in a common cluster are then gathered as an instance mask. This process is exemplified in Figure 2.9.

Getting neural networks to solve IS in a bottom-up fashion can be achieved through a wide variety of formulations. For instance, [NHD17, DBNMG17, FWR⁺17, KF18, KYL18, KL20] use abstract embeddings, while [UCFB16, KGC18, PZC⁺18, URF⁺18, YCZ⁺19, NBPG19, CCZ⁺20] build features related to coordinates in the image.

Among abstract embeddings, topologies can vary wildly: some methods, *e.g.*, [NHD17, DBNMG17, FWR⁺17] learn to output linear embeddings (unrestricted in $\mathbb{R}^D$), while some use different output channels to “color” non-overlapping instances in different mutually exclusive channels [KYL18]. Similarly, embeddings can be normalized on a sphere [KF18]. In all those cases, the assignment of an instance to a value emerges at training, *i.e.*, associatively. Hence, we refer to those as leveraging associative embeddings. This is not the case of [KL20] where the authors find a set of harmonic functions that provide sufficiently separated embeddings between instances



Figure 2.8: Top-down IS methods are prone to error in the presence of occlusions. (i) Bounding box outputs are too numerous and need to be filtered before segmentation, potentially omitting the detections of overlapping objects (merged detections, top) or leaving false positives (supernumerary detections, bottom). (ii) Ownership of bounding box can be ambiguous when one bounding-box covers multiple instances (top), (iii-iv) introducing noise in both the bounding box and semantic segmentation tasks. (v) Instance masks are limited to the spatial support of the estimated bounding boxes (top, bottom).

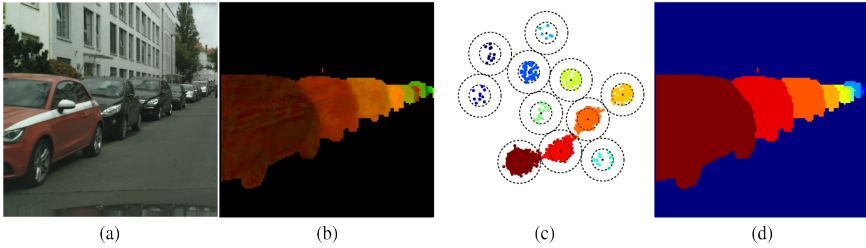


Figure 2.9: Illustration of a bottom-up method, from [DBNVG17]. The input image (a) is fed to a neural network that provides pixelwise embeddings, represented as colors in (b) and points in the embedding space in (c). A clustering is performed in the embedding space (c). Pixels corresponding to each cluster in the embedding space are regrouped in an instance mask (d).

is found prior to any training, and then teach the network to estimate their values.

Methods that use coordinate-related embedding space may simply estimate the coordinates of the center of mass of instances [KGC18, NALV18, NBPG19, CCZ⁺20], or the bounding box coordinates [YCZ⁺19]. In cases where pose is available, joint coordinates can be used as well [PZC⁺18]. We refer to them as leveraging spatial embeddings. As a counter-example, pixels can be classified according to the direction in which its center lies [UCFB16].

In this dissertation, we turn our attention to methods that are based on associative and spatial embeddings. We now introduce common concepts before exploring a small set of representative methods: associative embeddings [NHD17], spatial embeddings with fixed centroids and instance bandwidth [CCZ⁺20], and spatial embeddings with learnable centroids and instance bandwidth [NBPG19]. This list is far from exhaustive, but provides a representative overview of the many possibilities we unify in Chapter 5.

Foreground and background. Not all pixels belong to objects. IS is only useful in regions covered with countable, distinguishable objects to differentiate, *i.e.*, regions covered with semantic classes of *things*, which we regroup under the term *foreground* and set of pixels $\mathcal{F}^*$. In contrast, *stuff* classes denote the semantic classes in which objects cannot be distinguished, which we regroup under the term *background* along with the classes that are not of interest.

Network predictions and embeddings. Raw pixelwise network predictions $\Phi(p) \in \mathbb{R}^D$ are turned into embeddings $\Psi(p)$ lying in the method’s embedding space.

Instance centroid. To each instance i such that there exists an annotated instance mask $\mathcal{I}_i^*$ is associated an instance centroid o_i^* in the embedding space, most often considered as the average embedding in the instance.

Pull loss. The network is trained to make each foreground pixel’s embedding close to its instance centroid through a pull loss L_{pull} .

Push loss and learnable centroids. A push component L_{push} can be added to the loss to prevent centroids of different instances from collapsing into a single point. This is useful when centroids are *learnable*, *i.e.*, when centroid o_i^* depends on output embeddings $\Psi(p)$.

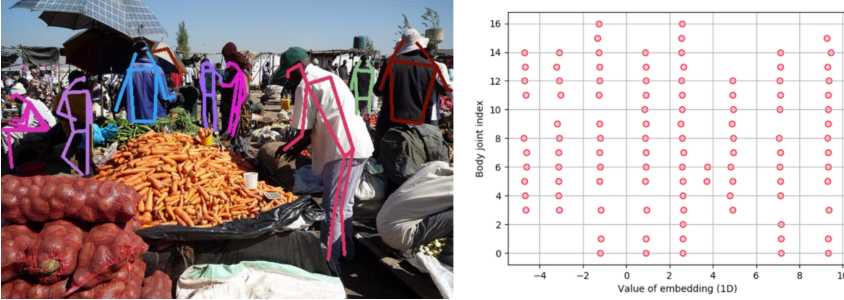
Seeds. At inference, a *seeding mechanism* provides candidate centroids $\mathcal{O} = \{o_j\}$ to perform the clustering. This usually relies on an additional map to be predicted by the network or on the histogram of predicted embeddings.

Clustering and predicted instance mask. The clustering associates each pixel p with its predicted instance label $I(p)$. This is usually performed by taking the closest candidate centroid in the embedding space, *i.e.*, $I(p) = \operatorname{argmin}_j \|o_j - \Psi(p)\|_2^2$. Each predicted instance mask can be extracted as $\mathcal{I}_j = \{p : I(p) = j\}$.

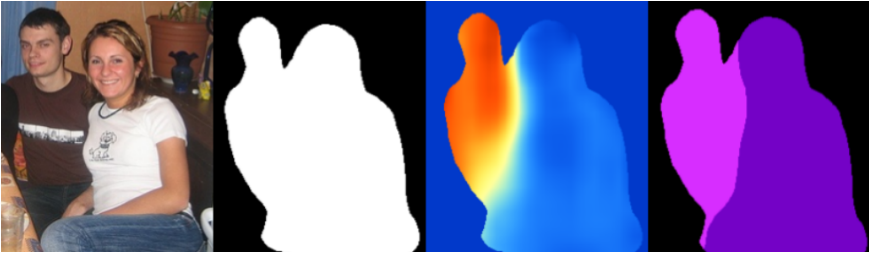
2.4.1 Associative embeddings

Newell *et al.* [NHD17] first introduce associative embeddings for pose estimation as features that encode the grouping of joints: joint detections with similar associative embedding should be grouped together. In the same work, the authors propose to use it for IS, although with unconvincing performance.

In parallel to foreground segmentation or keypoint detection, their model predicts 1-D embeddings $\Psi(p) \in \mathbb{R}$ that are equivalent to the network outputs $\Phi(p)$ and that have to be separated by a margin defined by the hyperparameter σ . Example embeddings and resulting groupings are illustrated in Figure 2.10 for both tasks. Each ground-truth instance centroid o_i^* is defined as the average embedding in its mask:



(a) Pose estimation example. Left: Image and skeletons obtained from grouping detected keypoints. Right: 1-D embedding space used to group keypoints.



(b) IS example. From left to right: image, foreground segmentation, 1-D pixel-wise embedding represented through color, predicted instance masks. While the embedding is relatively constant within each person, a transition region appears at their border.

Figure 2.10: Illustration of associative embeddings for pose estimation and IS, from [NHD17].

$$o_i^* = \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} \Psi(p) . \quad (2.12)$$

Network supervision. The task for the neural network is twofold: *(i)* each pixel in an instance should have an embedding close to its instance centroid, this requirement translates in a pull loss L_{pull} ; *(ii)* centroids of different instances should be far apart from each other, which translates in a push loss L_{push} .

The authors define L_{pull} and L_{push} by sampling a random set of pixels S_i for each ground-truth instance i , and applying attractive and repulsive forces among them:

$$L_{\text{pull}} = \frac{1}{|\mathcal{I}^*|} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{p \in S_i} \sum_{p' \in S_i} \|\Psi(p) - \Psi(p')\|^2 \quad (2.13)$$

$$L_{\text{push}} = \frac{1}{|\mathcal{I}^*|^2} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{j \neq i} \sum_{p \in S_i} \sum_{p' \in S_j} \exp\left\{-\frac{1}{2\sigma^2} \|\Psi(p) - \Psi(p')\|^2\right\} , \quad (2.14)$$

where σ is a fixed hyperparameter.

Inference. With low-dimensional and well separated embeddings, decoding is quite simple: obtaining the predicted centroids $\mathcal{O} = \{o_j\}$ only involves computing a histogram with a bin size that depends on σ followed by NMS. Each foreground pixel can then be assigned to its closest centroid:

$$I(p) = \underset{j}{\operatorname{argmin}} \|\Psi(p) - o_j\|^2 . \quad (2.15)$$

Limitations. Their experiments reveal that such a formulation is more competitive for pose estimation, *i.e.*, grouping a sparse set of keypoints, than it is for IS, *i.e.*, regrouping dense sets of points. Increasing the embedding dimension did not help.

Several challenges may explain their difficulties. In their framework, *(i)* 1-D embeddings are easy to cluster but they make it hard to avoid different instances from having similar embeddings, *(ii)* higher-dimensional embeddings make it easier to avoid similar embeddings but harder to decode clusters from the histogram, *(iii)* adjacent instance masks in IS inevitably imply

continuums in the embedding space, exacerbating collision issues in (i-ii), and may as well undermine the convergence of the associative embeddings through the minimization of the losses in Equations (2.13) and (2.14). (iv) As discussed in [NALV18], purely associative embeddings prevent the network from working locally, as different labels have to be assigned to similar instances appearing at different places. This causes the network to be less data- and parameter-efficient, as discussed in Section 2.2.2.

2.4.2 Spatial embeddings with Panoptic-Deeplab

In this dissertation, we often take as a starting point Panoptic-Deeplab [CCZ⁺20], a bottom-up method which is well known for its simplicity, efficiency, and reproducibility. As illustrated in Figure 2.11, Panoptic-Deeplab performs semantic segmentation through the output of semantic maps $\mathbf{C}(p, c)$, estimates offsets $\Phi(p)$ from instance pixels to their instance center of mass and estimates a heatmap $\mathbf{H}(p)$ indicating the presence of instance centers.

To each of those outputs corresponds a ground-truth: $\Phi^*(p)$ is the offset from p to its instance centroid; $\mathbf{H}^*(p)$ is a heatmap with a Gaussian kernel centered around each ground-truth centroid o_i^* , their width depending on an hyperparameter σ ; $\mathbf{C}^*(p, c)$ is a one-hot version of $\mathbf{C}^*(p)$:

$$\begin{cases} \mathbf{H}^*(p) = \max_i \exp\{-\frac{1}{2\sigma^2}\|p - o_i^*\|_2^2\} \\ \Phi^*(p) = o_{I^*(p)}^* - p \\ \mathbf{C}^*(p, c) = \mathbb{1}_{\{c=C^*(p)\}} \end{cases} \quad (2.16)$$

Network supervision. Panoptic-Deeplab is trained by minimizing the global loss function

$$L = \lambda_{\text{sem}} L_{\text{sem}} + \lambda_{\text{seed}} L_{\text{seed}} + \lambda_{\text{emb}} L_{\text{emb}} \ , \quad (2.17)$$

made from L_{sem} , L_{seed} , and L_{emb} that are, respectively, the semantic segmentation, detection heatmap, and embedding loss functions; with λ_{sem} , λ_{seed} , and λ_{emb} being their corresponding weights. It uses the weighted bootstrapped categorical cross entropy function introduced in [YCZ⁺19] for semantic segmentation loss, MSE function for the seed loss and L1 function

for the embedding loss:

$$\begin{cases} L_{\text{seed}} = \frac{1}{|P|} \sum_{p \in P} |\mathbf{H}(p) - \mathbf{H}^*(p)|^2 \\ L_{\text{emb}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \|\Phi(p) - \Phi^*(p)\|_1 \\ L_{\text{sem}} = -\frac{1}{|\Omega|} \sum_{p \in \Omega} \sum_{c=1}^K \mathbf{C}^*(p, c) \log(\mathbf{C}(p, c)) \end{cases} \quad (2.18)$$

where the subset Ω contains top- k pixels with highest L_{sem} loss. For L_{emb} , the average is taken over only the foreground (or thing) pixels.

Inference. Semantic class of each pixel is identified by computing $\mathbf{C}(p) = \operatorname{argmax}_{c \in \{1, \dots, K\}} \mathbf{C}(p, c)$. Then, the mask of foreground pixels $\mathcal{F}$ is built by regrouping the masks of object classes. Spatial embedding vectors are first computed from predicted offset vectors, *i.e.*, $\Psi(p) = \Phi(p) + p$. Instance centroids $\mathcal{O} = \{o_1, \dots, o_I\}$ are extracted by applying an NMS on the predicted heatmap [CCZ⁺20]:

$$\mathcal{O} = \text{NMS}(\mathbf{H}, P) . \quad (2.19)$$

The foreground pixels p are then grouped together with respect to their closest candidate centroid:

$$\mathcal{I}_i = \{p \in \mathcal{F} : \|o_i - \Psi(p)\|_2 < \|o_j - \Psi(p)\|_2, \forall j \neq i\} . \quad (2.20)$$

Limitations. With Panoptic-Deeplab’s formulation for spatial embeddings, all the difficulties raised in Section 2.4.1 are resolved: (i) fixed spatial centroids avoid collisions between remote objects naturally and (ii) seeds can be extracted through the simple spatial NMS from Equation (2.19). (iii) Continuums in the embedding space exist, but do not affect the seeding that is based on a seed map, nor the convergence of the embeddings as their ground-truth is static (iv) the network solves a translation-equivariant task as its role is to estimate a local offset from each instance pixel to its instance centroid.

Depending on the dataset, there may, however, be occurrences of very close centroids. This may (i) confuse the centroid predictions as the network may fail to output distinct centroids in the seed map for two or more overlapping instances, and (ii) supervision may fail to provide a significant difference between the embeddings of such overlapping instances, as the offsets would have very close targets. How can we therefore supervise the network to distinguish both instances, and decode them individually? In-

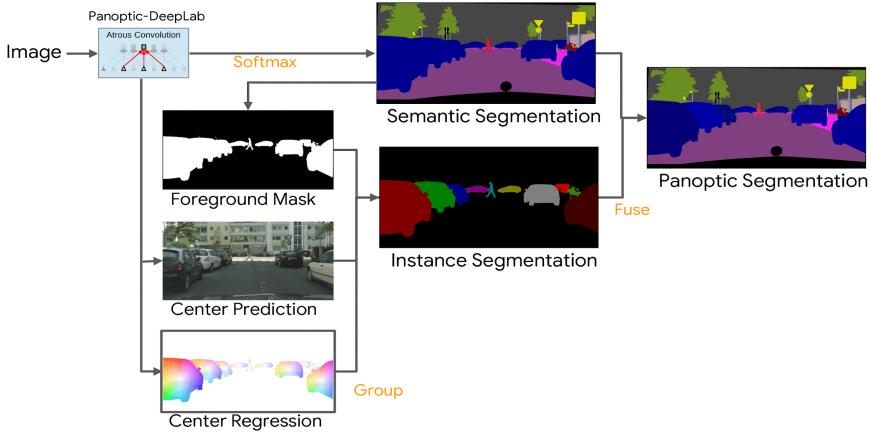


Figure 2.11: Illustration of the Panoptic-Deeplab method [CCZ⁺20] corresponding to spatial embeddings with fixed centroids. The network predicts the semantic segmentation, a heatmap with Gaussian kernels around instance centers of mass, and offsets from pixels to their instance centers. Pixels segmented with object classes are grouped to constitute the foreground. Each foreground pixel is assigned to a center depending on its offset, yielding to instance masks. Instance masks and semantic segmentation are fused to yield to the panoptic segmentation. Another illustration, specific to our work and notations, is available in Figure 6.2.

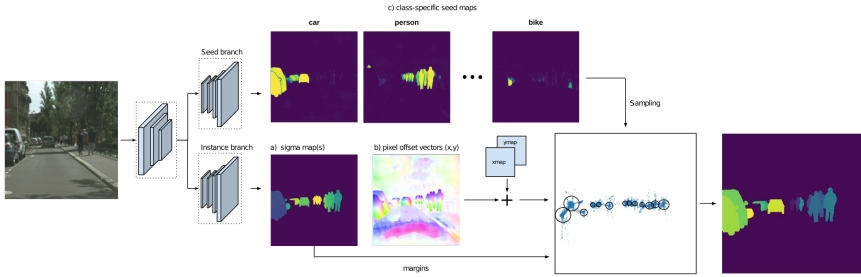


Figure 2.12: Illustration from the method proposed in [NBPG19], representative of spatial embeddings with learned centroids. Being fed with the input image X , offset from pixels to their ground-truth instance centroids and estimated quality of those offsets. The method also learns instance-wise bandwidth σ_i , which we do not integrate. Pixelwise embeddings are computed by adding offsets and pixel positions. Pixels that are estimated to have high-quality embeddings are clustered in the embedding space, yielding instance masks.

terestingly, as will be presented in the next section, solutions were known prior to Panoptic-Deeplab, but were not taken into account in its design, its authors preferring to favor simplicity and robustness of their solution.

2.4.3 Spatial embeddings with learnable centroids and instance bandwidth

Neven *et al.* [NBPG19] introduce several improvements on top of classical spatial embeddings. They propose, among other things, to learn the centroids and provide a particular loss to achieve this optimization. To this end, the centroids are formulated as those of associative embeddings, see Equation (2.12):

$$o_i^* = \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} \Psi(p) . \quad (2.12)$$

Network supervision. The authors supervise the prediction of embeddings by optimizing the Intersection-over-Union (IoU) between each ground-truth mask and the set of pixels that lie close to it in the embedding space. In practice, they define quality maps $m_i \in [0, 1]^P$ reflecting how close each pixel p is close to belonging to each instance i , according to the bandwidth

hyperparameter σ . The LovaszHinge function is a piecewise linear convex surrogate to the Jaccard loss, whose minimization maximizes the IoU between two masks [YB15, BTB18], and can directly be minimized between each instance quality map m_i and its binary ground-truth $m_i^* \in \{0, 1\}^P$:

$$\begin{cases} L_{\text{emb}} = \sum_{\mathcal{I}_i^* \in \mathcal{I}^*} \text{LovaszHinge}(m_i, m_i^*) \\ m_i(p) = \exp\{-\frac{1}{2\sigma^2} \|\Psi(p) - o_i^*\|_2^2\} \\ m_i^*(p) = \mathbb{1}_{\{p \in \mathcal{I}_i^*\}} \end{cases} \quad (2.21)$$

In this case, the loss maximizes the IoU between m_i and m_i^* , thus guiding $m_i(p)$ to 1 (resp., $\Psi(p)$ to o_i^*) for p inside the instance mask $\mathcal{I}_i^*$ and to 0 (resp., away from o_i^*) outside of it in a principled way.

As ground-truth centroids o_i^* are not fixed, *i.e.*, they are free to move across space since their definition depends on network predictions, the seed map proposed in Equation (2.16) may fail to be predicted properly by a CNN. Indeed, the CNN may not be able to associate the peaks to predict with a particular visual pattern anymore. Making the assumption that the network will have an easier time at predicting which pixels have high-quality embeddings, an elegant solution is to estimate a quality measure of the embedding in every pixel, *e.g.*, $m_{I^*}(p)$, through a pixelwise network output $\mathbf{H}(p)$ supervised by a regression loss L_{seed} [NBPG19]:

$$\begin{cases} \mathbf{H}^*(p) = \mathbb{1}_{\{p \in \mathcal{F}^*\}} m_{I^*}(p) \\ \mathbf{H}(p) \text{ a CNN output} \\ L_{\text{seed}} = \frac{1}{|P|} \sum_{p \in P} (\mathbf{H}(p) - \mathbf{H}^*(p))^2 \end{cases} \quad (2.22)$$

Inference. Interestingly, $\mathbf{H}(p)$ can be used to obtain an estimation of the foreground mask [NBPG19]. From the foreground mask, the decoding is performed iteratively: (i) the unassigned pixel s with the highest score $\mathbf{H}(s)$ is sampled, its embedding $\Psi(s)$ being taken as predicted centroid o_j . (ii) All unassigned foreground pixels p with $\exp\{-\frac{1}{2\sigma^2} \|\Psi(p) - o_j\|_2^2\} > 0.5$ are assigned to $\mathcal{I}_j$. (iii) Steps (i-ii) are repeated with incremented j as long as the unassigned pixel with the highest score is above a certain threshold t , so as to consider sufficiently high-quality seeds only.

This means that the extraction of the seeds and the pixel assignment to instance masks $\mathcal{I}_j$ are performed jointly. Untying both processes, the greedy

extraction of the seeds $\mathcal{O} = \{o_j\}$ can be seen as an NMS in the embedding space: (up to the implementation details discussed in Appendix A.1)

$$\mathcal{O} = \text{NMS}_{\psi \in \Psi} \mathbf{H}(\mathcal{P}(\psi)) , \quad (2.23)$$

with $\mathcal{P}(\psi)$ being the pixel proposing ψ and the NMS neighborhood being defined with respect to the instance bandwidth σ :

$$\mathcal{N}(\psi) = \{\psi' : \exp\{-\frac{1}{2\sigma^2}\|\psi' - \psi\|_2^2\} > 0.5\} . \quad (2.24)$$

Equivalent pixelwise instance labels $I(p)$ are obtained by selecting, for each pixel, the most confident seed prediction that lies sufficiently close to the pixel embedding:

$$I(p) = \underset{j: \exp\{-\frac{1}{2\sigma^2}\|o_j - \Psi(p)\|_2^2\} > 0.5}{\text{argmax}} \mathbf{H}(\mathcal{P}(o_j)) . \quad (2.25)$$

Limitations. As-is, the greedy clustering may fail to prevent undesired instances from being detected at the border of real instance masks because of the optimistic way the margin used. Apart from this, with the introduced mechanisms, Neven *et al.* offer in [NBPG19] answers to our questions from the previous section: spatial embeddings with learnable centroids can increase the distance between centroids so that the clusters are better distinguished. The seed map they propose, although potentially more difficult to post-process and perhaps less reliable, is more versatile, allowing to handle learned centroids and embeddings lying in other spaces than the image space. Those concepts are particularly useful for the general framework proposed in Chapter 5.

The paper goes a few steps further, learning instance-specific, horizontal and vertical instance bandwidth, *i.e.*, σ_{ix} and σ_{iy} . However, none of our experiments seemed to get the dynamics of the training right. The embeddings of remote parts, *e.g.*, human feet, failed to reach the instance centroids, which we could explain by gradients vanishing as the value of the m_i saturates. Additionally, predicted σ_i and seeds were not reliable enough to obtain robust clustering at inference. For those reasons, we chose to explore in the rest of this dissertation less complex methods, *i.e.*, simpler losses and constant σ taken as hyperparameter.

2.5 Neural architectures supporting bottom-up instance segmentation methods

In theory, any neural network architecture predicting an output vector for each pixel can be used in a bottom-up IS method. Therefore, semantic segmentation architectures, that label each pixel through a vector output, are natural candidates to tackle IS. Semantic segmentation is most naturally and efficiently solved using a CNN free of any fully connected layer, *i.e.*, in a fully convolutional manner [LSD15]. As we will see, however, some additional aspects are worth taking into account to ensure decent performance.

Solving IS, akin to semantic segmentation, requires a certain context, *i.e.*, a sufficient receptive field, around each pixel to label it correctly. In order to increase the receptive field quickly while using small, *e.g.*, 3×3 , convolution kernels, architectures usually start by downsampling the image by a factor of two every few convolutional layers, for instance with pooling operations, while increasing the number of features. We usually call *encoder* this section of the network, which computes a compressed representation of the image. The *decoder*, a second sequence of convolutional layers, follows the encoder to provide an output with the desired output resolution, *i.e.*, in our case, decoding pixelwise outputs from the compressed representation.

In essence, what we just described for the encoder is no different from what is performed by most networks solving image classification. As proposed by several works [LSD15, WSVDH19], a simple architecture for semantic segmentation, and, therefore, for IS [DBNVG17], can be obtained by taking a classification architecture, removing the last pooling and fully connected layers, and adding one or two convolutions for decoding. As reported in [WSVDH19], however, such simplistic architectures may lack some important features necessary for state-of-the-art performance.

First, modern semantic and instance segmentation architectures include shortcuts from high resolution layers in the encoder to high resolution layers in the decoder [RFB15], so that the high-resolution details do not have to be encoded in the low-resolution representation.

Second, semantic and instance segmentation architectures usually enforce the creation of multi-scale, or even global, features before the end of the encoder [ZSQ⁺17, CPSA17]. While already useful in semantic segmentation, large-scale features are particularly important in IS when some objects span a large portion of the image and all their pixels' outputs need to agree on a label. Global features are even more important when using

associative embeddings, as a global context is the only mechanism helping avoid embedding collisions between remote but similar-looking instances.

Architectures satisfying those requirements were already legion in the literature prior to 2019, as surveyed in Section 4.1.1. Because of all the possible combinations, the specificities of data and task formulations, getting any guarantee that a given architecture will provide the optimal performance on the problem at hand is untractable, even under some hardware constraints. In our work, we made the arbitrary but motivated choices to use the ICNet architecture [ZQS⁺18] in Chapter 4, and the Panoptic-Deeplab architecture [CCZ⁺20] in Chapters 5 and 6. Both satisfy the considerations mentioned above; ICNet was the architecture of a top-performing semantic segmentation method at the time of our work on team discrimination, preceding IS-specific architectures, while the Panoptic-Deeplab architecture is a specialized version of the Deeplabv3+ [CZP⁺18] semantic segmentation architecture that has been subject to heavy hyperparameter search for top bottom-up IS performance specifically. Additionally, at the time of this writing, the Panoptic-Deeplab method as a whole still provides a top-performing IS baseline.

2.6 Conclusion

Section 2.1 introduced the IS task, common datasets and metrics. We have described how multi-class datasets and metrics may fail to reflect the actual IS performance, which justifies performing single-class studies first.

Section 2.2 surveyed the main concepts behind CNNs for IS. We have seen that a CNN was made by stacking convolutional layers, and that convolutional layers were equivalent to dense layers sliding spatially. This has brought us to consider the fact that CNN translation-equivariance, *i.e.*, having the CNN to perform a local task, may allow for better efficiency with respect to the number parameters and samples used for training.

Section 2.3 introduced top-down IS methods, and reported technical weaknesses that impede good performance in case of occlusions. In short, top-down methods first perform bounding box detection, and only then segment the main object in each detection. In case of occlusions, regressed bounding boxes tend to be more noisy, easily cancelled, and to have more ambiguous targets, hence why they struggle to perform accurate IS.

Section 2.4 introduced general concepts that are common between bottom-up methods, and surveyed three representative methods: *(i)* Associative em-

beddings [NHD17], (ii) Spatial embeddings with fixed centroids [CCZ⁺20] and (iii) Spatial embeddings with learnable centroids and instance bandwidth [NBPG19]. The strengths and weaknesses of their different components have been discussed. To summarize, associative embeddings have difficulties to deal with many instances, large images, are tedious to train and decode, but can elegantly discriminate instances locally. Fixed spatial embeddings shine by their simplicity and robustness in all those aspects, except for their ability to solve occlusions which is limited when centroids get close to each other. Learnable instance centroids and bandwidths are sound mechanisms to add to spatial embeddings to help solve those cases, but result in less robust training and inference processes. Those considerations will help us gather robust components in a method capable of solving occlusions in Chapter 5.

Finally, Section 2.5 discussed key considerations to take into account when choosing or designing a network architecture to solve IS. We saw that semantic segmentation architectures, *e.g.*, ICNet [ZQS⁺18], used in Chapter 4, were natural candidates to address IS, but that architectures are now available in the literature whose hyperparameters have been optimized for IS performance, *e.g.*, that of Panoptic-Deeplab [CCZ⁺20], used in Chapters 5 and 6.

Chapter 3

Modular output mathematics

Broadly speaking, modularity is the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use. The concept of modularity is used primarily to reduce complexity by breaking a system into varying degrees of interdependence and independence across and "hide the complexity of each part behind an abstraction and interface".

– “Modularity”, Wikipedia

Modularity is a key aspect of research code bases, as there is value in trying different variants for one component and different sets of components. For this reason, modularity is pervasive in the Deep Learning (DL) community, many libraries proposing interchangeable components for most of the requirements of DL experiments. For instance, external libraries provide varieties of optimizers, datasets, data augmentation definitions that can be used off-the-shelf to perform experiments [Cho15, PGM⁺19]. This is also the case for the definition of neural networks architectures. In that case, residual blocks, convolutional or pooling layers can easily be combined and substituted to define network variants [Cho15, PGM⁺19], supporting the emergence of automatic search for neural architectures [JSH19].

In the rest of this dissertation, we use the term *Output Mathematics (OM)* to refer to the set of functions that compute losses, the functions

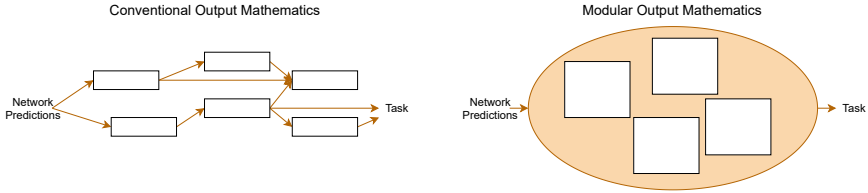


Figure 3.1: Implementation of OM: (left) Traditionally, OM are implemented as a set of functions whose calls have to be carefully orchestrated by external code because of the flow of data enforced by their signatures, as depicted by orange arrows. (right) Our solution consists in splitting the OM into classes in object-oriented programming. Those are assembled using multiple inheritance to create the OM component. Variables of interest to the OM are accessed through class methods, so that they become available everywhere in the OM, without requiring intricate signatures and dependencies. This is depicted by the orange pool, in which modules can freely access any data.

required to translate the network output into semantically meaningful information, and potentially many other functions that compute intermediate variables or are required to visualize the manipulated concepts (*e.g.*, accuracy or quality metrics, but also high-dimension tensors visualized based on a PCA transform), for debugging purposes, for instance.

As we study a wide variety of interdependent supervisions and decoding systems for Instance Segmentation (IS) through the thesis, *e.g.*, see Figure 5.3 for the main components implemented in Chapter 5, the need for a modular way to handle those mathematics becomes overwhelming. This need is, however, not specific to IS, and is most likely to be felt whenever the dependencies of the interdependent components are not consistent across all the experiments or are subject to change.

OM are generally not modular because they are conventionally implemented as a set of functional abstractions (*i.e.*, functions). Fundamentally, functions are constrained by their signatures, *i.e.*, their list of arguments, and are thus especially suited to deal with scenarios where the programmer wants to test multiple versions of a single function whose interface with the rest of the program is fixed. Interdependencies therefore require careful orchestration to feed each function with the right arguments. Hence, as depicted in Figure 3.1 (left), the functions written in the code base of such experiments end up in being largely constrained by their signatures, which determine the way they access the variables they process. They depend on

each other and are not plug-and-play: the alternatives to a function are limited to those with the same interface, which can be designed to be called at the same time with the same dependencies. Consequently, the changes in dependencies induced by slight modifications in the task formulation might impact the code in many places. This hampers the deployment of a modular implementation paradigm, rendering existing OM methods more difficult to reuse and alternative task formulations more difficult to explore. Furthermore, promoting modularity keeps sub-tasks free to evolve in unforeseen ways at minimal code rewriting cost, and help comparing alternative task formulations by pointing explicitly the modules that differ between them.

In this chapter, we therefore propose a first practical solution to make the OM modular, and illustrate how the underlying structure of such a modular framework favors fluent research, enabling easier increments, more accurate comparisons and a more maintainable code base.

Our prime contribution lies in a set of recommendations to facilitate researchers' work when exploring the space of formulations to address a given task. As illustrated in Figure 3.1 (right), the key idea of our proposal consists in making the OM variables accessible without constraining function signature. This is achieved by defining the OM modules as classes, and by making all OM variables accessible through dedicated methods. The multiple inheritance principle of classes, with method overriding and mixins, are shown to offer a natural way to increment the code or explore variants of the problem formulation. Our proposal, grounded in the class inheritance paradigm, is a step forward in offering modularity, with minimal constraints on functions interfaces and inter-dependencies. By relaxing the assumptions made about how those functions interact with the rest of the code (both in terms of access to data and function composition), it facilitates the presentation of the OM as an inheritance diagram that explicitly presents how methods differ from a baseline, with the purpose of facilitating both the transfer of knowledge and the incremental development of DL models.

In a nutshell, through appropriate definition of classes and hierarchical dependencies, we show that multiple inheritance offers the flexibility needed to overwrite and/or compose functions at will, making the OM module easy to adapt or substitute by another.

The rest of this chapter is organized as follows. Section 3.1 surveys the inheritance mechanisms that are mobilized to implement our solution. It also concretely defines the notion of OM. Section 3.2 presents our proposed paradigm to implement the OM as a modular component, and explains why and how this paradigm overcomes most of the issues encountered by the con-

ventional and scattered implementation alternatives. Section 3.3 illustrates our proposal on the Panoptic-Deeplab IS method [CCZ⁺20], demonstrating the ease of integration, added flexibility, and negligible overhead for DL tasks. Section 3.4 concludes.

3.1 Background

3.1.1 Network output mathematics

Fundamentally, in a DL model, the OM correspond to the set of operations that are related to the transformation of the output activations of a neural network into semantically meaningful information. They are defined in mathematical language, through the manipulation of mathematical variables. They include (i) the definition of the training loss(es), typically comparing the network output(s) with some representation of the desired prediction(s), but also (ii) the functions that process/decode the (multiple) outputs of the network to turn them into meaningful predictions.

For example, modern pose estimation methods, *e.g.*, [KBA19], predict a set of heatmaps in which hot-spots locate a specific type of keypoint (*e.g.*, head or feet), together with vector fields that are used to associate the key-points of an instance. As another example, popular IS methods predict a seed map to locate the instance center, together with a map of embeddings, defined so that the pixels of individual instances aggregate in tight and distinct clusters in the embedding space [NHD17, NBPG19]. In this formulation, each instance center corresponds to a hot-spot in the seed map, and the pixels of a specific instance are simply identified as the ones whose embedding lies sufficiently close to the embedding of that particular instance centers. Figure 3.2 illustrates this principle in the case where the embedding space corresponds to 2D spatial coordinates, defined in each pixel as the sum between the pixel coordinates and a 2D offset that is predicted by the network. In this scenario, the network is trained to make the embeddings of all pixels associated to a same instance equal to a common 2D vector, referred to as the instance center. This corresponds to the Panoptic-Deeplab method, defined in [CCZ⁺20]. The figure presents the main variables involved in the IS process. The variables depicted in blue on the left-side are directly predicted by the network, in the form of raw tensors. Other variables are computed from the dataset (ground-truth column), from the raw network predictions (predicted maps column), or from both the dataset

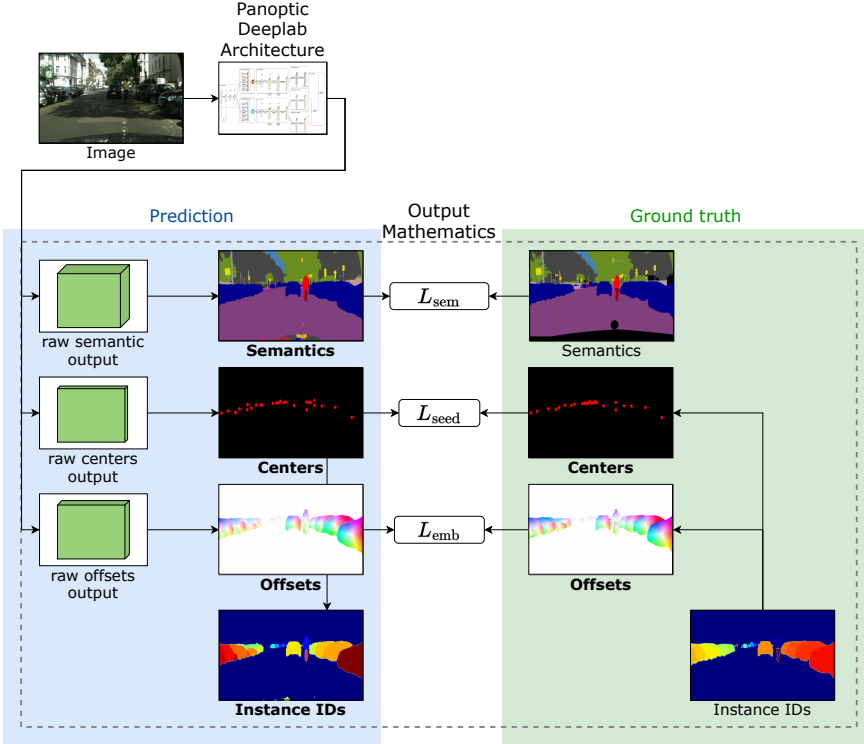


Figure 3.2: Diagram of the Panoptic-Deeplab [CCZ⁺20] method illustrating its mathematical variables and their flow. Some of those mathematical variables are provided by the dataset (*e.g.*, image, ground-truth instance labels), some are provided by the output of the neural network (*e.g.*, raw semantic, centers and offsets outputs), while the other variables, indicated in bold, have to be computed from those for training the model or interpreting its results. We propose to regroup all the variables inside classes so as to make them easy to access, interchange and modify, see Figure 3.6.

and predicted maps (loss functions). The mathematical operations that are used to compute all those variables (*i.e.*, maps and losses) correspond to the so-called OM, and directly reflect the computer vision problem formulation. In Section 3.2, we present a number of implementation guidelines that offer the flexibility required to change those functions at will. The application of those guidelines to implement and compare variants of the Panoptic-Deeplab formulation is presented in Section 3.3 to demonstrate the advantages of our proposal in terms of incremental development and knowledge transfer. Since those guidelines build on the multiple inheritance paradigm, we briefly review its key principles in the next section.

3.1.2 Multiple inheritance

Fundamentally, in object-oriented programming, classes are extensible templates for creating objects, providing the initial values for object instance variables, and the bodies for methods [Bru02]. All objects generated from the same class share the same methods, but manipulate distinct instance variables. New objects are created from a class by calling its constructor with specific arguments.

A class can inherit the base behavior of another class. Its methods are defined by default to those of its base class, but they can also be overridden to refine their behavior as needed. For compatibility reasons, the overriding method usually has the same name, parameters, and return type as the method it overrides. Overriding helps in avoiding code duplication between different classes that would only differ through subtle variants of their methods.

For this reason, classes are a construct of choice when engineering modular systems. A base class can define the generic behavior of a component as well as how it should interact with the external world, while leaving some methods to be refined by its child classes, created for a specific purpose. In the early age of object-oriented programming, it was not well understood how a class should behave if it had to inherit from several classes implementing a same method differently. For this reason, and because it proved to be sufficient to design most systems, many languages only allowed single inheritance, and engineers learned to design systems that way.

However, as illustrated in Figure 3.3, multiple inheritance largely increases object-oriented programming flexibility. It enables an object or class (here, D and E) to inherit characteristics and features from more than one parent (here B and C). To address the ambiguity as to which parent a par-

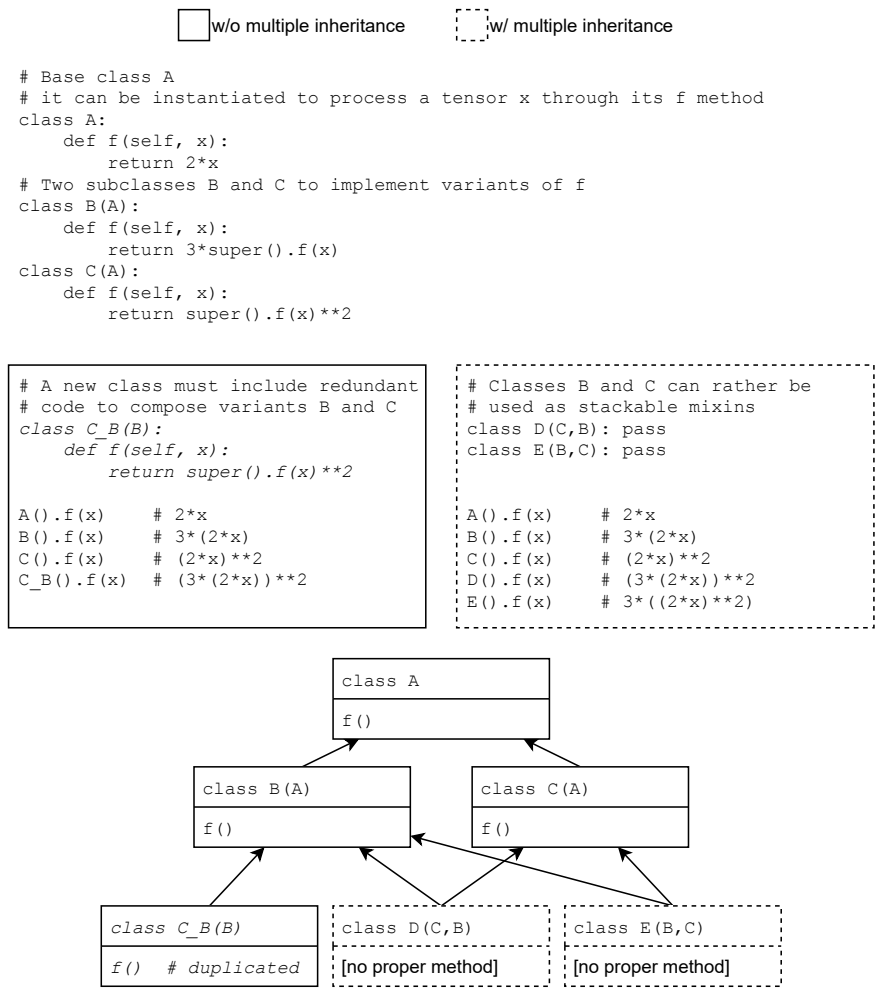


Figure 3.3: An example of the flexibility brought by multiple inheritance. Single inheritance does not make it trivial to create a module with B and C features at the same time. One solution could imply to duplicate the code of one in a child class of the other (as done here by C_B), but this raises maintenance issues. In contrast, multiple inheritance enables the mixing of both features in a new class without requiring additional code, nor structurally complex code.

ticular method (here f) is inherited from when more than one parent class implements the said method, composition mechanisms have been proposed. In Python, the order of inheritance affects the class semantics, and the ambiguity is solved using the C3 linearization (or Method Resolution Order (MRO)) algorithm [BCH⁺96]. This gives the language the opportunity to support multiple inheritance with consistent and well-understood behavior.

This solution to disambiguate multiple inheritance makes it possible to exploit *mixins* [BC90], which denote classes (*e.g.*, B and C in Figure 3.3) that are defined with the purpose of being combined with a base class. The combination of a mixin with a set of base classes results in a family of modified classes (like D and E in Figure 3.3). It allows a programmer to share a functionality between multiple classes without duplicating the same code in them. An example is illustrated in Figure 3.3 where a single mixin (B or C) is created to affect some methods (here f) in a precise way without having to know exactly into which base class it will be combined.

When a class inherits from several mixins, the modifications induced by the mixins are composed according to the order in which the mixins are listed in the inheritance declaration. See Figure 3.3 for examples.

3.2 Modular implementation of output mathematics

We here propose a solution to manipulate the network OM as a modular component. Our solution aims at promoting reuse and incremental refinement, across experiments, of the functions involved in the mathematical formulation of the task handled by the network. It is expected to support incremental and experimental development, but also to facilitate the exploration of various formulations of the optimization problem addressed by the training phase, *e.g.*, based on different losses. Those alternative formulations (*e.g.*, of the computer vision problem of interest) might even consider the prediction of alternative variables than the ones considered in the baseline reference formulation.

Our proposal should be considered as a reference baseline, to be challenged and refined in future research. It aims at enabling fluent composition and replacement of OM methods, making it possible to share hands-on experiments, and to compare between distinct formulations of the relation between the network prediction and the task at hand. In practice, our solution consists of a number of implementation guidelines that can be summarized

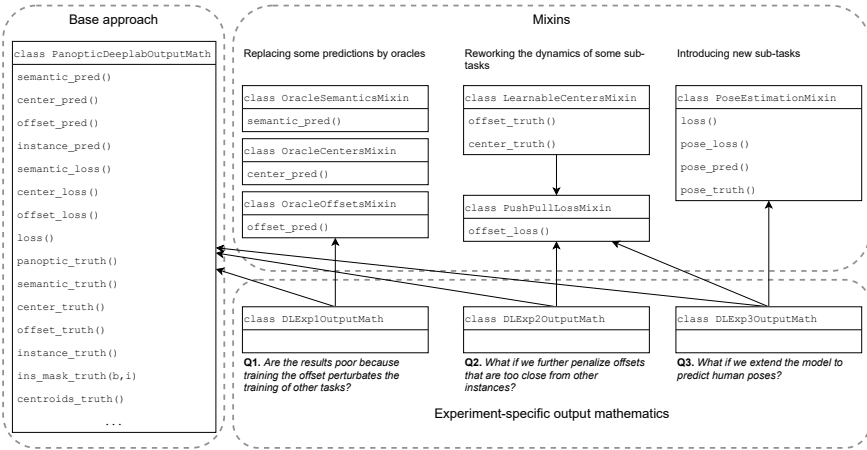


Figure 3.4: A use case in which our two recommendations are applied to make the OM flexible and to make its code reusable across many formulations. The same code, augmented with a few mixins, is used to investigate 3 questions (Q1, Q2, and Q3) related to the behavior or the extension of a baseline, without having to perform any modification to the code bases of experiments (except maybe for additional heads of the neural architecture in the case of an additional sub-task, which are most often modular anyhow).

as follows.

Our first recommendation consists in assigning a specific class to each DL experiment, *e.g.*, to each problem formulation. Each of those classes should host as methods the mathematical functions implied in the computation of the task results and associated losses. To minimize code duplication, each class should inherit from mixins the behaviors that deviate from a baseline formulation. The baseline should thus be defined to implement methods shared across multiple modified experiments. This is illustrated in Figure 3.4 for the use case introduced in Figure 3.2, and further studied in Section 3.3.

The second recommendation addresses the data accessibility. To reduce the constraints and inter-dependencies induced by the inclusion of arguments in the signature of methods, the variables that are relevant to the OM, including ground-truth, raw predictions, and high-level variables derived from the latter, should be part of the object instantiated from the OM class and be made accessible through its methods with as few parameters as possible. In that way, those variables become available to all the other methods of the object, without having to anticipate their usage within a list of arguments in the signature of each method. This offers an increased flexibility when reusing or overriding a method.

This flexibility is illustrated in Figure 3.5, which depicts an example where a method is overridden by the child class. On the left side, each method receives a particular variable as parameter, which causes an issue when the child class method wants to use another variable than the one considered in the base class. In contrast, on the right side, all variables that are relevant to the studied problem are part of the class' instance variables and are available through dedicated methods, without the need to be passed as arguments. As a consequence, the child class method can be adapted whatever the variable required by its new behavior. In more detail, the goal here is to create a mixin replacing the predicted semantic segmentation by the ground-truth semantic segmentation. Such replacements can be useful for ablation studies for instance, to reevaluate a metric in case a sub-task is perfectly solved, or to evaluate whether the training of a particular sub-task perturbs that of others. In this situation, we can observe on the left side that the ground-truth segmentation cannot be returned by the `OracleSegmentationMixin` simply because the ground truth has not been routed as an argument of the inherited `semantic_pred` method. Indeed, based on the base class, external code is only providing to the `semantic_pred` method the `semantic_activations` parameter, not

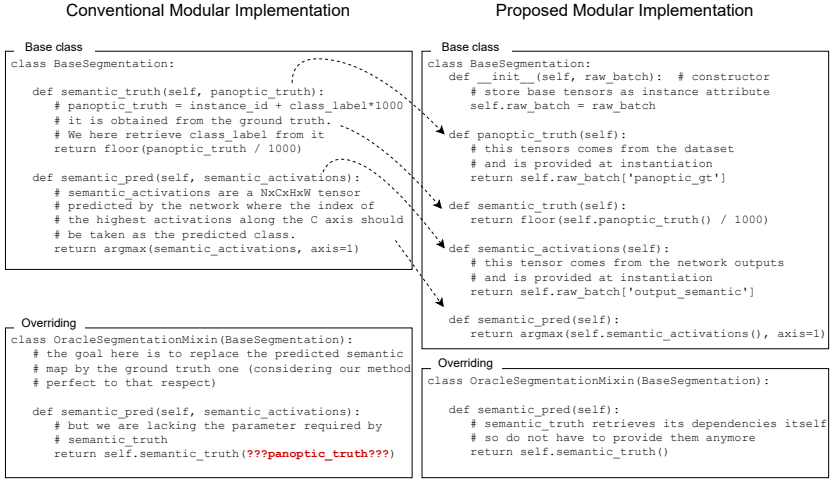


Figure 3.5: (left) Situation where the traditional parameter passing of variables prevents a mixin to implement a desired feature. The goal here is to create a mixin replacing the predicted semantic segmentation by the ground-truth semantic segmentation. The `semantic_pred` and `semantic_truth` methods respectively receive the `semantic_activations` and `panoptic_truth` parameter. Therefore, mixins cannot expect their methods to receive other parameters. It makes it impossible for `OracleSegmentationMixin` to redefine `semantic_pred` so that it returns the `semantic_truth` variable. (right) By making variables accessible through methods calls, passing variables as function arguments is not necessary anymore. This reduces the constraints imposed by the method signatures and renders the above-described modification trivial.

the `panoptic_truth` required by the `semantic_truth`. It is not rare to encounter similar situations in less trivial examples: predicting uncertainties, integrating new tasks in the losses, or defining a variable as a function of the network output rather than as a constant derived from the dataset ground truth. This last case occurs for example when replacing the constant ground-truth instance center by the mean of the embeddings associated to the ground-truth instance [NHD17, NBPG19].

To avoid duplication of computation, we add a memoization mechanism that caches the result on the first call and restitutes it on subsequent calls. Systematic caching means that any variable computed on a batch stays in memory for as long as that batch. Though this may sound naive, it often makes sense in the context of deep model training since most intermediate results (among which neural network activations) have to be remembered for the backpropagation anyway. Those activations alone usually outweigh by a large margin most of the byproducts required to compute the loss. Furthermore, not caching a variable in the OM may result in computing it many times, which potentially also leads the backpropagation library to store it multiple times, thereby increasing the memory-footprint.

It can however make sense to allow the programmer to introduce exceptions to this rule. Such a mechanism has been implemented in the example code provided with this work [Ist21].

Section 3.3 considers a representative example of a computer vision DL algorithm to challenge our assumptions about the limited memory and computational impact induced by our proposed implementation paradigm on training performance. Beyond showing that the overhead is far from prohibitive, this assessment first reveals that our proposed recommendations for modular OM can be naturally implemented in the reference Panoptic-Deeplab method, with minimal changes to the reference code. Second, the modular OM implementation is leveraged to prototype a variation of the IS problem formulation, incorporating learnable instance centers in the instance prediction process. This work reveals that, thanks to its modularity, the OM component is straightforward to adapt to the new formulation of the problem.

3.3 Assessment on the Panoptic-Deeplab method

Panoptic-Deeplab is a pixelwise panoptic segmentation approach that, although surprisingly simple, achieves state of the art performance. It is presented in more details in Section 2.4.2.

This use case has been chosen because it fits perfectly the scope of this dissertation and provides a popular and representative DL solution to a fundamental computer vision problem. The integration of our modular component to this case is released in the form of a complete repository in [Ist21]. It first reveals that an arbitrary code can conveniently be adapted to host our proposed modular and composable implementation of the OM, with limited resource overhead. It also illustrates how the flexibility offered by our solution facilitates documentation of the incremental development of an alternative formulation of the IS problem.

To demonstrate the feasibility and the advantages of our proposal, we proceed in two steps. First, we apply our recommendations for modular OM on the public Panoptic-Deeplab reimplementation. We observe that this can be done easily, with minimal changes to the reference code. Second, we exploit our modular OM to prototype a variation of the IS formulation, incorporating learnable instance centers in the instance prediction process. Figure 3.6 illustrates how we have chosen to formulate the Panoptic-Deeplab method using modular OM and to extend them for the learnable centers.

Time and memory overheads, induced by the use of our modular OM component, are estimated, with and without the added learnable centers functionality, and appear to be negligible in comparison to the added flexibility and achievable optimizations.

3.3.1 Modular output mathematics

Integrating our recommendations for a modular implementation of the OM into the public Panoptic-Deeplab implementation [CCZ⁺20] is fairly straightforward. Details about our specific implementation choices can be found in the code [Ist21]. Here, we survey the main steps involved during this development.

The first step consists in implementing the four upper classes illustrated in Figure 3.6, associating a class method to every mathematical concept present in the initial code. Sometimes, a same concept can be represented

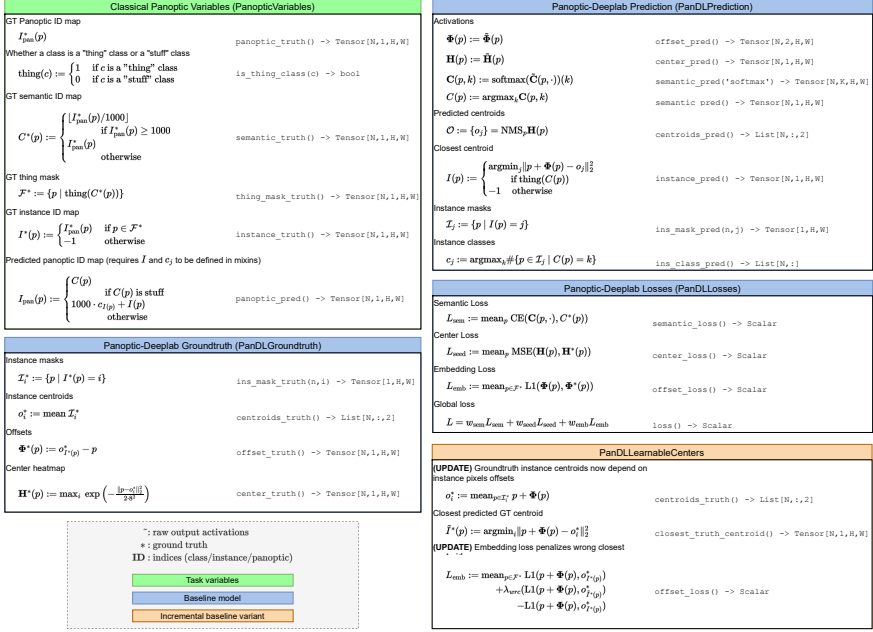


Figure 3.6: Illustration of an implementation of the baseline Panoptic-Deeplab OM and of its incremental extension (to support learnable centers), following our recommendation to exploit multiple inheritance for modularity. The different blocks, devoted to variables and OM functions definitions, are classes that can be inherited by the final OM of an experiment. Those blocks consist in a set of lines that each associates a variable introduced to formulate the task to its corresponding method in the code. The green block includes task specific variables, *i.e.*, variables that are relevant whatever the mathematical formulation. Blue blocks present the variables manipulated by the baseline reference method, while the orange block introduces the novel variables complementing the baseline with learnable centers.

in different ways and it is difficult to come up with different names. In such cases, it can be practical to handle the different representations of a concept in a method with a “representation” parameter. For instance, the semantic maps are predicted by the network as unbounded activations maps, from which probabilities can be obtained via the application of the softmax function, and the predicted class through the argmax function. We use the representation parameter to refer to them as `semantic_pred('logits')`, `semantic_pred('softmax')` and `semantic_pred('argmax')` respectively.

The OM class is in our case an anonymous class created once and for all at the beginning of the execution, inheriting from `PanDLBase`, itself inheriting from the four aforementioned classes, and the eventual mixins specified by the practitioner in the experiment configuration. Each raw batch is used as a parameter to the constructor in charge of instantiating the class to create an instance specific to that set of data.

Each obtained instance can then be forwarded to the parts of the code requiring access to tensors, *e.g.*, to perform the optimization step, to compute and log metrics, or to draw visualizations. In our case, we reused the argument dedicated to the raw batch. Finally, data computations in those parts of the code only need to be replaced by calls to the adequate methods of our OM instance. This way, those components that refer to the output variables do not contain code that is specific to Panoptic-Deeplab anymore. Hence, they become compatible with alternative implementations of the OM.

To handle the caching, we make the `PanDLBase` class also inherits from the `OutputMath` class that implements the default memoization policy described in Section 3.2 while providing ways to specify exceptions to it.

3.3.2 Implementing the LearnableCenters mixin

Formulating the IS problem based on learnable instance centers implies that the network is not explicitly supervised to predict centroids at the center of instances anymore. Instead, each ground-truth instance center location depends on the output of the network, and get positioned at the center of mass of the locations $p + \Phi(p)$ pointed pixels of that instance. This way, the network is given the freedom to redistribute instance centers spatially, which may help to solve clustering ambiguities when some instances have centers that are close to each other. This idea is common in the literature [NHD17, NBPG19].

Were the ground-truth offsets and center heatmap computations im-

plemented conventionally, adjusting them would have required heavy rearranging of the code and its data flow. In the public Panoptic-Deeplab reimplementation for instance, those targets are computed on the CPU before sending data to the GPU and running the neural network, and therefore have no access to the latter’s predictions.

In our case, it appears, as suggested in Figure 3.6, that only overriding the method associated with the o_i^* variable is required. For better centroid separation, however, we also choose to adapt the offset loss with a term that penalizes pixels pointing to locations $p + \Phi(p)$ lending closer to other centroids than that of their ground-truth instance.

In the next section, we assess that the overhead introduced by our modular OM is not prohibitive with respect to their benefits.

3.3.3 Overhead induced by the modular framework

Four experiments have been run and compared: *(i)* the original Panoptic-Deeplab training loop, where the target maps (*i.e.*, ground-truth centers, offsets, and semantic labels) are computed on the CPU; *(ii)* an equivalent training, with our proposed modular OM but accessing the targets maps that are still computed on the CPU; *(iii)* a training with target maps being computed in the OM component, on the GPU; and *(iv)* a training with the learnable centers mixin replacing the fixed centers in the OM component.

In each of those, we track the duration of forward pass, backward pass, both happening on the GPU, and the overall time of a training batch, which includes computing the targets in CPU threads when needed. We are also interested in the maximal GPU memory allocated, since it generally constrains the batch size.

We perform those experiments on a machine with one NVidia V100 GPU, a batch size of 4, and dedicate 4 CPU cores to data loading (of an Intel Xeon Gold 5217). All the other parameters come from the MobileNetV2 configuration of Panoptic-Deeplab. The results are averaged over 100 consecutive batches.

Table 3.1 reports each metric for all experiments. The only difference between *(i)* and *(ii)* is the integration of the OM component. A significant amount of computations still happen outside the OM because the target maps are computed by the data-loader CPU threads. Both rows exhibit very similar timings and peak memory usage, which confirms that the overhead introduced by the component, its calls and caching mechanisms, is negligible. Implementation *(iii)* moves the computation of targets from

Experiment	Forward pass (s)	Backward pass (s)	Overall batch time (s)	Peak memory (GB)
Original Panoptic-Deeplab	0.290 ± 0.004	0.471 ± 0.019	5.370 ± 1.566	26.230 ± 0.000
Panoptic-Deeplab w/ ModulOM	0.293 ± 0.009	0.474 ± 0.053	5.390 ± 1.476	26.246 ± 0.001
Panoptic-Deeplab w/ ModulOM w/ targets in OM	0.387 ± 0.029	0.477 ± 0.008	1.010 ± 0.099	26.640 ± 0.043
Panoptic-Deeplab w/ ModulOM w/ targets in OM w/ learnable centers	0.667 ± 0.655	0.556 ± 0.030	1.500 ± 0.819	27.427 ± 0.301

Table 3.1: Average ($\pm$ standard deviation) duration of the forward pass, the backward pass, overall batch and the peak GPU memory usage over 100 batches. Four scenarios were considered: the original method with the original code, the original method with a shallow integration of our OM component (the targets are still computed on the CPU), the same with target computation integrated in the modular OM (on the GPU), and a variant of the method with learnable instance centers.

data-loading happening in CPU threads to the OM component. The overall batch processing time is heavily reduced compared to *(ii)*, because less data is required to be transferred between CPU and GPU, and because the GPU is a more efficient device to perform those operations, despite the many loops that have to take place. Would it not be the case, it would still be possible for the programmer to specify that some computations inside the OM have to take place on the CPU. We emphasize that, as suggested in Section 3.2, despite the systematic caching of intermediate results taking place, the increase in peak memory is relatively low. Finally, *(iv)* adds the learnable centers on top of *(iii)*. The overhead of this particular variant is significant because it fundamentally impacts the mathematical system, using network predictions for creating targets, hence creating new operations contributing to the gradients. In addition, the computation of the new embedding loss sometimes has to occur on the CPU because computing the distance between the embedding of each pixel and each ground-truth instance centroid requires more memory than available on the GPU¹. Yet, all-in-all, the modularity of our OM allows us to implement it in a way that it is still on average more than three times faster than the public Panoptic-Deeplab

¹Since then, implementation tricks have been found to lift this restriction, resulting in better performance of learnable centers, therefore not invalidating the results.

implementation.

We see the fact that this complexity could be added seamlessly, contained in a class definition, as an example that the proposed modular OM are effective to broaden the possibilities of existing code bases in a practical fashion.

3.4 Conclusion

This chapter has proposed to group the functions implementing the OM inside a modular component, offering increased flexibility when designing DL experiments, by allowing easy substitution and modification of the mathematical functions associated to the problem formulation handled by the neural network.

Research on OM, as we perform in Chapters 5 and 6, would otherwise require constant refactoring of code that is characterized by a large number of interdependent functions, dealing both with network- and task-related data. This is in contrast to most other DL components, that are related either to the network (*e.g.*, optimizer, scheduler, or network layers) or to the task (*e.g.*, data augmentation or task-related metrics and visualization routines), and are characterized by well-defined interfaces that make their functional abstraction relatively straightforward.

Our modular implementation of the OM builds on two main recommendations that can be summarized as follows. First, by implementing the OM as methods associated to a set of base and mixin classes, various problem formulations can be designed in a modular way by leveraging the multiple inheritance paradigm. Second, by exposing the mathematical variables involved in the problem formulation through methods with only necessary parameters, those variables become available to all other methods, relaxing the constraints and dependencies imposed by method signatures over the code and the data flow.

Application of those guidelines to the Panoptic-Deeplab method has revealed that our proposed modular implementation is easy to integrate in an existing experimental code base, without penalizing its performance. The flexibility of the proposed approach has also been demonstrated by showing how easy it is to change the problem formulation, the definition of associated variables and the data flow.

This work provides a convenient way for practitioners to run the many different experiments involved in the development of DL models, and facili-

tates the comparison of similar formulations. It therefore answers our needs for Chapters 5 and 6. Moreover, because it gives access to new formulations from a limited amount of code, it could open perspectives regarding the automatic exploration and optimization of the OM, as is done for some other parts of the DL experiments such as the data-augmentation pipeline and the neural architecture [CZM⁺19, JSH19].

Chapter 4

Team segmentation using pixelwise associative embedding

This chapter presents a use case of Instance Segmentation (IS) where the concept of instance is extended to bags of instances with similar appearances. Specifically, in each image, each entity to segment consists in a set of objects that share similar visual appearance. This extension is suited to the formulation of team discrimination in team sports scenes, a problem that was not addressed using deep learning prior to this work.

It is worth mentioning that since its publication, in 2019, related approaches have emerged but are not discussed, *e.g.*, [KPE21], and that this work could benefit from the results of Chapters 5 and 6 to enhance supervision and decoding. In the light of our complete work, it may also seem weird that we did not benchmark spatial embeddings. As the text will reflect, at first, we did not envision team segmentation outside of the associative embedding framework, and introduced it with the hope of seeing a visual descriptor emerge. With our current knowledge, we appreciate that spatial embeddings could be used to tackle this task, but believe that they would be subject to limitations that associative embeddings do not have: the team centroids may collide in unintuitive cases, *e.g.*, (i) when teams are spread evenly or (ii) when the estimation of the center of mass of each team is subject to noise, for instance when the membership of one or more players' is uncertain. Additionally, in team discrimination, (iii) spatial embeddings

cannot work locally anymore, while *(iv)* long-range spatial embeddings are known to be less precise [PZC⁺18]. For those reasons, we believe that associative embeddings provide a more natural formulation for Convolutional Neural Networks (CNNs) to solve that problem.

Team sports analytics has numerous applications, ranging from broadcast content enrichment to game statistical analysis for coaches [CLC⁺16, TGM⁺17, ZYH16]. Assigning team labels to detected players is of particular interest when investigating the relationship between team positioning and sport action success/failure statistics [BLC⁺14, HPSL18, LC14], but also for some specific tasks such as offside detection in soccer [DLS⁺09] or ball ownership prediction in basketball [WSL⁺15].

Many previous works have investigated computer vision methods to detect and track team sport players [CDI⁺19, DLS⁺09, LCLH18, LTLM13, MEAM17, PDV17, TLWZ11]. They can detect individual players, but generally resort to unpractical manual intervention or to unreliable heuristics to adapt their processing pipeline to recognize the players' team. Specifically, they generally need human intervention to adjust the team discriminant features (*e.g.*, RGB histogram in [LTLM13], or CNN features in [LCLH18]) to the game at hand [BLC⁺14, LC14, LCLH18, LTLM13]. A few methods have attempted to derive game-specific team features in an automatic manner [DLS⁺09, TLWZ11]. They consider the unsupervised clustering of color histograms [DLS⁺09] or bags of color features [TLWZ11] computed on the spatial support of the players that are detected in the game at hand. Those methods depend on how well color discriminates the two teams, but is also quite sensitive to occlusions and to the quality of player detection and segmentation [MEAM17]. This probably explains why those previous works have been demonstrated in outdoor and highly contrasted scenes, as encountered in soccer for example. We show in Section 4.4 that those methods fail to address real-life indoor cases.

As observed in [LCLH18], indoor sports analytics have to deal with lower color contrast between players and background, and more dynamic scenes, with more frequent occlusions. [PDV16, PDV17] also point out the low illumination, the strong reflections induced by dynamic advertising boards, the severe shadows, the large player density and the lack of color discrimination in indoor scenes.

In this work, we use the approach of associative embeddings, presented in Section 2.4.1, that was introduced to discriminate instances in grouping problems [ND17, NHD17]. However, differently from [ND17, NHD17], it is demonstrated using a lightweight ICNet CNN (opening broader deploy-

ment perspectives than the heavy stacked hourglass architecture promoted in [ND17, NHD17]) and, to our knowledge, is the first work assigning similar embeddings to unconnected pixels, thereby extending the field of application of pixelwise associative embedding. Thanks to associative embeddings, the features to discriminate the teams are learnt from data, and do not require game-specific adjustment. Associative embeddings consist a generic learning-based strategy that aims at predicting a feature vector in each pixel, in such a way that, independently of the game at hand, similar vectors are predicted in pixels lying in players from a same team, while distinct vectors are assigned to pairs of pixels that correspond to distinct teams. In other words, we train a neural network to separate, in an embedding space, the pixels of different teams and to group those in common team. A simple and efficient clustering algorithm can then be used to dissociate different teams in an image. Hence, we do not rely on explicit recognition of specific teams, but rather learn how to map player pixels to a feature space that promotes team clustering, whatever the team appearance. Although teams change at each game, there is thus no need for fine tuning or specific manual annotation for new games.

To validate our method, we have trained our network on a representative set of images captured in a variety of games and arenas. Since only a few player keypoints (head, pelvis, and feet) have been annotated in addition to the player team index, the player segmentation component of our network has been trained with approximate ground-truth masks, corresponding to ellipses connecting the key points. Our CNN model is validated on games (teams) and arenas that have not been seen during training. It achieves above 90% team recognition accuracy, despite the challenging scenes (indoor, dynamic background, and low contrast) and the inaccurate segmentation ground-truth considered during training. Interestingly, the lightweight backbone makes the solution realistic for real-time deployment.

This chapter is organized as follow. Section 4.1 reviews the related works associated to CNN-based sport analysis, segmentation, and associative embedding. Section 4.2 then introduces our proposed method, using a ICNet variant to both segment the players and compute pixelwise team discriminant embeddings. The experiments presented in Section 4.4 demonstrate the relevance of our approach, while conclusions and some perspectives are provided in Section 4.5.

4.1 Related works

Recent developments in computer vision make an extensive use of CNNs [RDS⁺15]. This section reviews the specific type of CNNs, named Fully Convolutional Network (FCN), that is used for image segmentation at the time of this research. It then introduces the recent associative embedding methods considered to turn object class segmentation into object IS.

4.1.1 Fully Convolutional Network (FCN)

Fully Convolutional Networks are characterized by the fact that they output spatial feature maps, strictly computed by the sequential application of convolutional layers, generally completed with ReLU activation and batch-normalization or dropout regularization layers.

In recent works dealing with sport video analysis, FCNs have been considered for specific segmentation tasks, including player jersey number extraction [GLM17], soccer field lines and players segmentation [CDVD18]. In [LCLH18], a two-steps architecture, inspired by [YCL16] and [YK16], is even proposed to extract players bounding-boxes with team labels. The network, however, needs to be trained on a game-per-game basis, which is impractical for large scale deployment. None of these works is thus able to differentiate player teams without requiring a dedicated training for each game, as proposed in Section 4.2, where a real-time amenable FCN provides the player segmentation mask, as well as a pixelwise team-discriminant feature vector.

There are two main categories of real-time FCNs: encoder-decoder networks and multi-scale networks.

Encoder-decoder architectures adopt the encoder structure of classification networks, but replace their dense classification layers by convolutional layers that upsample and convolve the coded features up to pixelwise resolution.

SegNet (Segmentation Network) [BKC17] was the first segmentation architecture to reach near real-time inference. It is a symmetrical encoder-decoder network, with skip connection of pooling indices from encoder layers to decoder layers. ENet (Efficient Neural Network) [PCKC16] follows SegNet, but comes with various improvements, whose most prominent one is the use of a smaller decoder than the encoder.

Quite recently, several authors proposed to adopt multi-scale architectures to better balance accuracy and inference complexity. Considering

multiple scales allows to exploit both a large receptive field and a fine image resolution, with a reduced number of network layers. Among those networks, ICNet (Image Cascade Network) [ZQS⁺18] is based on PSPNet (Pyramid Scene Parsing Network) [ZSQ⁺17], a state-of-the-art network for non real-time segmentation. ICNet encodes the features at three scales. The coarsest branch is a PSPNet, while finer ones are lighter networks, allowing to infer segmentation in real-time. Two-columns network [WSvdH17], BiSeNet (Bilateral Segmentation Network) [YWP⁺18], GUN (Guided Upsampling Network) [Maz18] and ContextNet [PBLZ18] are composed of two branches.

4.1.2 Associative embedding

An embedding vector denotes a local descriptor that characterizes a signal locally in a way that can support a task of interest. Embeddings are thus not defined a priori. Instead, they are defined in an indirect manner, to support the task of interest. In computer vision, FCNs have recently been considered to compute pixelwise embeddings in a variety of contexts related to pixel clustering or pixel association tasks. In this context, FCN training is not supervised to output a specified value. Rather, FCN training supervises the relations between the embedded vectors, and checks that they are consistent with the task of interest.

In [VSF⁺18], the embedding vector is used to compute the similarity between two pixel neighborhoods from two distinct images, typically to support a tracking task. Interestingly, a proxy task that consists in predicting the (known) color of a target frame based on the color in a reference frame is used to supervise the training of the FCN computing the embeddings. Good embeddings indeed result in relevant pixel associations, and in accurate color predictions. This reveals that a FCN can be trained in an indirect manner to support various higher-level tasks based on richer pixelwise embedding.

Of special interest with respect to our team discrimination problem, *associative* embeddings have been introduced in [ND17, NHD17] and used in [LD18, ND17, NHD17] to associate pixels sharing a common semantic property, namely the fact that they belong to the same object instance. Authors in [NHD17] introduced associative embedding in the context of multi-person pose estimation from joints detection and grouping, and extended it to IS. More recently, [LD18] proposed CornerNet, a new state-of-the art one-shot bounding box object detector, by using associative embedding to

group top-left and bottom-right box corners. In all these publications, the network is trained to give close embeddings to pixels from the same instance and distant embeddings to pixels corresponding to different instances. All these works are based on the same heavy stacked hourglass architecture. However, [ND17] suggest that the approach is not strictly restricted to this architecture, as long as two important properties are fulfilled: first, the network should have access both to global and local information; second, pixelwise prediction at fine resolution is recommended, in order to avoid that a vector is subject to concurrent instances. This makes ICNet a premium candidate to segment players and compute team-specific embeddings in real time, since it computes features at three scales instead of two for other lightweight multi-branch FCN architectures.

4.2 Team segmentation using pixelwise associative embedding

Player team discrimination is not a conventional semantic segmentation problem since the visual specificities of each class are not known in advance. This section explains how associative embedding can be combined with player segmentation to address this problem.

4.2.1 Team discrimination & player segmentation

We propose to adopt the associative embedding paradigm presented in Section 2.4.1 to support the team discrimination task. In short, we design an FCN so that, in addition to player segmentation, it outputs, for each pixel, a D -dimensional feature vector that is similar for pixels that correspond to players of the same team, while being distinct for pixels associated to distinct teams. As explained in the previous section, embeddings learning is not based on an explicit supervision. Instead, embeddings are envisioned as a latent pixelwise representation, trained to support a pixelwise association task, typically to group [LD18] or match [VSF⁺18] pixels together. In the context of object detection, associative embedding has been applied with success in [LD18, NHD17] to group pixels corresponding to a same object instance. In these works, multiple hourglass-shaped networks are stacked recursively in order to progressively refine the 1-D embedding value that aims to differentiate object instances in a given class. Our work differs from [NHD17, ND17] and [LD18] in two main aspects.

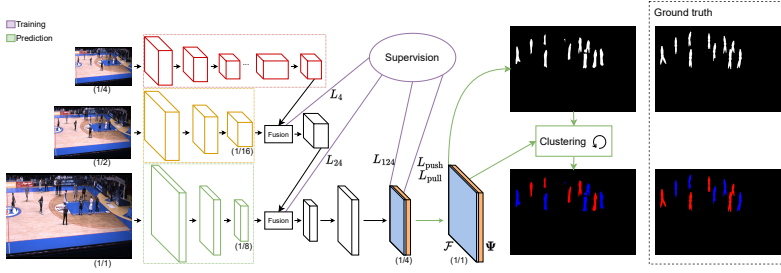


Figure 4.1: Overview of our architecture. ICNet [ZQS⁺18] is used as backbone for following assets: pixelwise segmentation, combination of three scales to encode global and local features, and speed (the original paper [ZQS⁺18] reaches 30 FPS at 1024×2048 resolution). Its last convolution layer is modified to output a segmentation mask along with vector embeddings in each pixel. We keep the multi-scale supervision for the segmentation and add L_{push} and L_{pull} losses to obtain similar embeddings in pixels of the same team and distant embeddings for pixels of different teams. During inference, a simple clustering algorithm can effectively split in different teams the pixels p from the segmentation mask $\mathcal{F}$ according to their embeddings $\Psi(p)$.

First, and because we target real-time deployment, the stacked hour-glass architecture is replaced by an ICNet [ZQS⁺18] backbone, as illustrated in Figure 4.1. As stated in [ZQS⁺18], ICNet reaches 30 FPS for images of 1024×2048 pixels on one Titan X GPU card. We use ICNet because its multi-scale encoders, along with a spatial pyramidal pooling, give access to a reasonably large receptive field (important to spatially share embedding information) while preserving the opportunity to locally exploit high-resolution image signal (important for a fine characterization of the content).

Second, our work deals with the problem of associating pixels of players that are scattered across the whole image. This is in contrast with the association of neighboring/connected pixels generally considered in traditional association tasks [LD18, NHD17].

4.3 Network architecture and supervision

The ICNet network architecture has mostly been left unchanged. Only the final convolution layer has been adapted to provide $D + 1$ channels. Those

comprise 1 channel for semantic segmentation, with a sigmoid activation, along with D channels for embeddings with linear activation. Figure 4.1 presents the player segmentation channel in blue while the D channels for embeddings are represented in orange. A number of loss functions are combined to train the network. Along with the multi-scale semantic segmentation loss from [ZQS⁺18], composed by L_{124} , L_{24} and L_4 , we add an embedding loss inspired by [NHD17, ND17, LD18]. It comprises two components, L_{pull} and L_{push} , which respectively pull teammates embeddings together and push opponents embeddings away from each other. L_{pull} and L_{push} losses only apply to the finest resolution. We have defined all loss components based on mean square distances.

For each scale $s \in \{124, 24, 4\}$, the semantic segmentation loss L_s is defined as the mean square distance between pixelwise foreground labels and predictions:

$$L_s = \frac{1}{H_s W_s} \sum_p |\mathcal{F}_s(p) - \mathcal{F}_s^*(p)|^2, \quad (4.1)$$

with H_s and W_s being the height and width of maps predicted at scale s , while $\mathcal{F}_s$ and $\mathcal{F}_s^*$ respectively denote the predicted and ground-truth player masks at scale s . Similarly, L_{pull} is formulated as the mean square distance between pixel embeddings and their centroids:

$$L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{\mathcal{I}_i^*} \sum_{p \in \mathcal{I}_i^*} \|\Psi(p) - o_i^*\|_2^2, \quad (4.2)$$

where o_i^* is the mean of the embeddings $\Psi(p)$ predicted over the pixels of team i , *i.e.*, $o_i^* = \sum_{p \in \mathcal{I}_i^*} \Psi(p)$.

In [NHD17], the push loss is expressed as a mean over pairs of pixels of a cost function that is chosen to be high (low) when pixels that are not supposed to receive the same embedding have a similar (different) embedding. Recently, [ND17] and [LD18] employed a "margin-based penalty", and wrote that this is the most reliable formulation they tested. Hence, we also adopt a margin-based penalty loss. Formally, L_{push} is defined similarly to L_{pull} , except that rather than penalizing embeddings that are far away from their centroid, it penalizes embeddings that are too close from the centroid of another team:

$$L_{\text{push}} = \frac{1}{|P|} \sum_{p \in \mathcal{F}^*} \sum_{\mathcal{I}_j^* \neq \mathcal{I}_{I^*(p)}^*} |1 - \|\Psi(p) - o_j^*\|_2^2|_+. \quad (4.3)$$

Our global objective function sums the contributions from the previously introduced losses:

$$L = \lambda_{124}L_{124} + \lambda_{24}L_{24} + \lambda_4L_4 + \lambda_{\text{pull}}L_{\text{pull}} + \lambda_{\text{push}}L_{\text{push}} , \quad (4.4)$$

with the loss weights having to be tuned (chosen values are explained in Section 4.3.1).

At inference, upsampling of last layer is inserted before activation (respectively, bilinear and nearest neighbor interpolations for segmentation and embedding channels). Then, a clustering algorithm is required to group pixels in teams. Fortunately, as observed in [NHD17], the network does a great job at separating the embeddings for distinct teams, so that a simple and greedy method such as the one detailed in Algorithm 1 is able to handle the clustering properly. As appears from the pseudocode, our naive clustering algorithm relies on the assumption that a player pixel embedding surrounded by similar embeddings is representative of its team embedding. Given a team embedding vector, player pixels are likely to be assigned to that team if their embedding lies in a sphere of radius 1 around the team embedding. We incorporate a refinement step in which we compute the centroid of the selected pixels. Then, to resolve ambiguities, player pixels are associated to the closest of the centroids.

4.3.1 Implementation details and hyperparameters

Our network is trained to extract players only, and to estimate associative embeddings for team discrimination. Referees and other non-player persons are part of the background class. Our work is based on the PyTorch ICNet implementation [Sha17]. Parameters have been empirically tuned. For the training, we employ Adam optimizer [KB15]. Loss weights defined in Equation 4.4 are: $\lambda_{124} = 1$ and $\lambda_{24} = \lambda_4 = 0.4$ as in original ICNet [ZQS⁺18], $\lambda_{\text{pull}} = \lambda_{\text{push}} = 4$ and are thus very different than the ones in [NHD17, ND17, ZQS⁺18] because our pull and push losses' definitions are averaged over pixels rather than over instances. Our best found learning rate is $lr = 10^{-3}$, and has been implemented with the "poly" learning rate decay taken from [ZSQ⁺17, ZQS⁺18, YWP⁺18] and their own sources. Compared to them, we apply the decay by epochs instead of iterations, but we keep the same power of 0.9. Hence, the learning rate at k^{th} epoch is $lr \cdot (1 - \frac{k}{max})^{power}$,

Input : $\mathcal{F}$ the predicted segmentation mask
 $\Psi(p)$ the predicted embedding in pixel p
Result: $I \in \llbracket 0; 2 \rrbracket^{H \times W}$ teams occupancy map (0 is associated to the background)

$\mathcal{W}(p)$ neighborhood of pixel p
 $S_p \leftarrow \frac{1}{|\mathcal{W}(p)|} \sum_{p' \in \mathcal{W}(p)} \|\Psi(p) - \Psi(p')\|_2^2$
 $j \leftarrow 0$ the team counter
 $\mathcal{R} \leftarrow \mathcal{F}$ the pixels to cluster
 $O \leftarrow D_1 \leftarrow D_2 \leftarrow \mathbf{0}^{H \times W}$
while $\mathcal{R} \neq \{\}$ **and** $n < 2$ **do**

$j \leftarrow j + 1$
 $p \leftarrow \operatorname{argmin}_{p \in \mathcal{R}} S_p$
 $o_j \leftarrow \Psi(p)$ the centroid for team j
 $\mathcal{M}_j \leftarrow \{p \in \mathcal{R} \mid \|\Psi(p) - o_j\|_2^2 < 1\}$
 $o_j \leftarrow \frac{1}{|\mathcal{M}_j|} \sum_{p \in \mathcal{M}_j} \Psi(p)$
 $\mathcal{M}_j \leftarrow \{p \in \mathcal{R} \mid \|\Psi(p) - o_j\|_2^2 < 1\}$
 $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{M}_j$
for $p \in \mathcal{F}$ **do**

$D_j(p) \leftarrow \|\Psi(p) - o_j\|_2^2$

end

end
for $p \in \mathcal{F}$ **do**

if $n = 1$ **then**

$I(p) \leftarrow 1$

else if $n = 2$ **then**

$I(p) \leftarrow \operatorname{argmin}(\{D_1(p); D_2(p)\}) + 1$

end

end

Algorithm 1: Fast clustering algorithm of pixels in the space of their associated embeddings. Up to two centroids are searched and refined, from the observation that for a team, embeddings of neighbour pixels can serve as initial prototype when they are similar. Embeddings similarity in the neighborhood of pixel p , $\mathcal{W}(p)$, is called S_p . After that, points are clustered according to their embedding vector's distance to centroids mapped in D_n arrays. This algorithm will inspire the Embedding Steadiness Seed Map (ESSM) introduced in Section 5.2.

with $max = 200$ denoting the total number of epochs, and lr being the base learning rate defined above. All but last layers of ICNet are initialized with pretrained Cityscapes ([COR⁺16]) weights from [ZQS⁺18], but a full training is done as the point of view adopted for sport field coverage is too different from the frontal point of view considered by cars in Cityscapes. Minibatch size is 16 and batch-normalization is applied. Neither weight decay regularization, nor dropout are added, but the following random data augmentation is considered: mirror flipping, central rotation of maximum 10 degrees, scaling such that $\min(\text{width}, \text{height}) = \text{random}([\frac{2}{3}, \frac{3}{2}]) \times 512$, color jitter in the perceptually uniform CIE L*C*h color space fixed to $L \pm 10$, $C \pm 7$ and $h \pm 30$ degrees, to keep natural colors. We trained the network on crops of 512×512 pixels, located randomly in scaled images. Validation is performed on 512×512 pixels patches, extracted from images scaled such as its $\min(\text{width}, \text{height})$ equals 512. For each model, we select the parameters of the best epoch according to a validation score defined as the mean of intersection over union of the two teams, between prediction and our approximate reference masks. Inference for testing is done on court images downsampled to 1024×512 and padded to preserve the aspect ratio.

In our implementation, we adopted 5-D embeddings, mainly because more dimensions a priori get more ability to capture/encode visual team characteristics unambiguously. We expect this ability to become especially useful when the receptive field does not cover the whole scene. In that case, the embedding prediction in one pixel may not be able to rely on a teammate appearance or on the absence of collision with an opponent embedding when those players are far and disconnected from the pixel of interest. The embeddings have thus to be consistent across the scene, despite their relatively local receptive field. In other words, they have to capture local team characteristics unambiguously. In practice, ICNet builds a global receptive field, and our trials provided similar results with 1- to 5-D embeddings.

4.4 Experimental validation

To assess our method, this section first introduces an original dataset, and associated evaluation metrics. It then runs a K-fold cross-validation procedure, and compares the performance of our associative embedding team discrimination, with a conventional color histogram clustering, applied on top of IS.

4.4.1 Dataset characteristics

To demonstrate our solution, we have considered an early version of the DeepSport dataset [ZSI⁺22] introduced in Section 2.1.2. It involves a large variety of games and sport halls: images come from 40 different games and 27 different arenas. Images show innumerable situations: occlusions between teammates and/or opponents, regular player distribution, absence or presence of all the players, images from training sessions and professional games with public, various game actions, still and moving players, presence of referees, managers, mascots, dynamic led advertisements, photographers or other humans, various lighting conditions, different image sizes (smaller dimension is generally close or superior to 1000 pixels). This dataset is composed of 648 images covering a bit more than half of the sport field. Each player has been manually annotated. Annotations considered in our work include a team label (Team A vs. Team B), and an approximate player mask. This mask has been derived from manual annotation of head, pelvis, and feet. It consists in seven ellipses approximately covering the head, the body (between head and pelvis), the pelvis, the legs (between pelvis and each foot), and the feet. Occlusions between ellipses of players located at different depth has been taken into account. Similarly to [CDVD18], our experiments reveal that the network can learn despite the coarseness of the masks. Players size in images feeding the network (scaling strategy in Section 4.3.1) is around $25 \times 75 \pm 15$ pixels.

4.4.2 Evaluation metrics

Our network enables player semantic segmentation, as well as team discrimination. Evaluation metrics should thus reflect whether players have been properly detected, and whether teammates have received the same team label. Therefore, we consider the following counters and metrics, to be computed on a set of test images:

- N_{miss} : Number of missing players
- N_{corr} : Number of correct team associations
- N_{err} : Number of incorrect team associations
- Missed players rate,

$$R_{miss} = \frac{N_{miss}}{N_{corr} + N_{err} + N_{miss}} \quad (4.5)$$

- Correct team assignments rate,

$$R_{CTA} = \frac{N_{corr}}{N_{corr} + N_{err}} \quad (4.6)$$

We now explain how the outputs of our network, namely the player segmentation mask and the map of team labels derived from the embeddings clusters, are turned into those evaluation metrics¹. Given a reference segmentation mask and a team label for each player instance, a simple majority vote strategy is adopted. A player is considered to be detected when the majority of pixels in the player IS mask are part of the segmentation mask predicted by the network. In that case, the majority label observed in the instance mask defines the team of the player. In practice, since our ground-truth mask only provides a rough approximation of the actual player instance silhouette, we resort to the part of the instance mask that is the most relevant for team classification, *i.e.*, to the two ellipses that respectively cover the body and the pelvis area. Since pixels that are in the central part of the body and pelvis ellipses are less likely to be part of the background, only the pixels that are sufficiently close to the main principal axis of the body/pelvis shape are considered. (A distance threshold equal to one third of the maximal distance between ellipse border and principal axis has been adopted. Changing this threshold does not significantly impact the results.)

4.4.3 Results

In order to validate the proposed team discrimination method with available data, we consider a K-fold cross-validation framework. It partitions the 648-images dataset into K disjoint subsets, named folds. Each K-fold iteration preserves one fold for the test, and use the other folds for training and validation. Average and standard deviation metrics can then be computed based on the K iterations of the training/testing procedure. In our case, ten folds of approximately equal size have been considered. Moreover, to assess whether the model generalizes properly on new games and new arenas, we construct the folds so that each fold contains images from distinct games and/or arenas. Table 4.1 lists cross-game folds characteristics, and Table 4.2 cross-arena folds characteristics.

¹Since accurate ground-truth segmentation masks are not available from the dataset (see Section 4.4.1), the segmentation quality can not be assessed based on conventional intersection over union metrics.

Fold	1 .. 3	4	5 .. 9	10
Train	516	518	520	518
Val	66	64	64	66
Test	66	66	64	64

Table 4.1: Splits of the dataset used for cross-game validation. Each column corresponds to one (set of) folds, and lines define the number of training/validation/test samples. Validation and test sets contain the images from 4 games.

Fold	1	2 .. 5	6	7	8 .. 9	10
Train	514	516	518	522	524	518
Val	66	66	64	62	62	68
Test	68	66	66	64	62	62

Table 4.2: Splits of the dataset used for cross-arena validation. Each column corresponds to one (set of) folds, and lines define the number of training/validation/test samples. Validation and test sets contain the images from 2 or 3 halls.

To estimate the value to give to our results, we compare them to a baseline reference. Since most previous methods recognize teams based on color histograms [DLS⁺09, LTLM13, TLWZ11], generally after team-specific training, we compare associative embeddings to a method that collects color histograms on player instances, before clustering them into two sets. In practice, as for the associative embedding evaluation, only the player pixels that are sufficiently close to the body/pelvis principal axis are considered to build the histogram in RGB, with 8 bins per dimension (512-dimensional histogram). Adopted clustering is the [PVG⁺11] implementation of variational inference algorithm with a Dirichlet process prior [BJ06], to fit at max two Gaussians representing our two clusters (two teams). This method has the advantage of being able to automatically reduce the number of prototypes, it is useful when less than two teams are visible in an image.

Results of cross-game validation are presented in Table 4.3, while cross-validation on sport halls is presented in Table 4.4. Standard deviations are low, demonstrating the weak dependence to a specific set of training data. Rate of missing detections is about 11%, which is an acceptable rate considering our backbone is the real-time ICNet model [ZQS⁺18] with arduous indoor basketball images. It could probably be improved with a finer tuning of hyperparameters, as well as more accurate segmentation

Method	R_{miss}	R_{CTA}
Associative Embedding	0.11 ± 0.04	0.91 ± 0.04
Color Histogram		0.62 ± 0.02

Table 4.3: Evaluation measures on cross-game K-fold: mean and standard deviation of missed player detection and of correct team assignment rates, for 10 folds.

Method	R_{miss}	R_{CTA}
Associative Embedding	0.11 ± 0.06	0.91 ± 0.03
Color Histogram		0.63 ± 0.02

Table 4.4: Evaluation measures on cross-arena K-fold: mean and standard deviation of missed player detection and of correct team assignment rates, for 10 folds.

masks and a formulation that involves a class for referees (see failure cases analysis below). More recent and effective improved segmentation networks could also be considered as long as they are compatible with associative embedding.

In Figure 4.2, we observe that players are generally well detected but roughly segmented, probably due to our approximate training masks. However, segmentation masks are very clean compared to the background-subtracted foreground masks derived for such kind of scenes (see for example [PDV17]). Therefore, they could advantageously replace those masks in algorithms using camera calibration to detect individual players from the segmentation mask [AJBV11, CSM12, DDD09].

In terms of team assignment, [LCLH18] mentions that they can not achieve good cross-game team assignment without fine-tuning. In comparison, our method reaches more than 90% of correct team assignments while testing on games and sport halls that are not seen during training. The baseline Bayesian color histogram clustering only reaches 62% of correct team assignments, which confirms that the team assignment task in the context of indoor sport is extremely difficult, as described in the introduction of this chapter. We get near identical results for cross-arena evaluation.

Qualitative results are shown in Figure 4.2. As written in Section 4.3.1, we intend to extract players only, excluding referees and other humans. Images belong to testing folds, meaning that they originate from games or arenas not seen during training. Teams masks are drawn in red and blue.



Figure 4.2: Team discrimination with associative embedding. From left to right: test image, zoomed reference masks and prediction. The first five rows present success cases, while the last two show failure cases. From top to bottom: running players; strong shadows; occlusions; court and teams share the same colors; only one team; confusion between players and referees; extreme occlusions. Please refer to the numerical version of the dissertation for the colors and ability to zoom on details.

The first five rows in Figure 4.2 illustrate how well the proposed method can deal with indoor basketball conditions. Players in fast movement and low contrast are detected and well grouped in teams. Occlusions, led advertisements, and artificial lighting are not a major problem. Associative embedding has a low sensitivity to high color similarities between background and foreground. Specific treacherous scenes with players of only one team and some other humans are correctly handled.

In addition to correctly segmented instances, our method sometimes predicts extraneous segments. We estimate the quantity of those additional segments to correspond to 10% of the number of annotated players. These detections come from referees and other unwanted persons on or close to the ground, and in certain cases from scenery elements. In basketball, the proportion of the number of referees related to the players is from 20 to 30% (we usually count 2 or 3 referees in a complete field, while players are $5 + 5$). Thus, it is interesting to see that our CNN trained on players generally avoids referees and other people. However, this is a challenging task, as can be seen in the two prominent failure cases shown in the last two rows of Figure 4.2, where referees shirts or pants are visually similar to a team. In the first example, a referee is detected as a player and included in a team (referee on the right, under the basket), and a player is filtered from predicted player class probably because it is seen as a referee by the network (background player in side of a referee). In the second example, the dark pants of a referee and a coach in the back of the court are assimilated to the team in black. This sample also presents a severe occlusion implying four players; inside and around this area, detection is inaccurate and team assignment of the orange player mixed with black teammates is lost.

4.5 Conclusion

In this chapter, we considered associative embedding to address the team assignment problem in team sport competitions. It offers the advantage of discriminating teams in sport scenes, without requiring an unpractical per-game training. Promising results are obtained on a challenging basketball dataset, with few tuning and only approximate player mask annotations. In comparison with classification [LCLH18], the generalization ability of the method is striking, despite the embeddings coming from a relatively simple multi-scale CNN, as opposed to the stacked hourglass network considered in previous works [LD18, ND17, NHD17]. That disconnected regions across

the whole image have to agree on a common embedding does not seem to cause any problem. The success of the method reveals that the notion of IS can be extended to more sophisticated tasks than individual object silhouette delineation. Although it will not be explored more extensively in this thesis, this suggests that the IS paradigm can be used to solve other problems, *e.g.*, to track instances across frames or for action spotting, and that associative embeddings should be able to achieve long-range agreement, *i.e.*, reconnections.

Chapter 5

Associative versus spatial embeddings for instance segmentation

In 2018, the literature of Instance Segmentation (IS) has shifted from associative embeddings [NHD17, LLW⁺17, FWR⁺17, DBNVG17, KF18] to spatial embeddings in a striking fashion [KGC18, NALV18, PZC⁺18, URF⁺18, YCZ⁺19, NBPG19, CCZ⁺20]. As we have seen in Section 2.4, those families of methods are very similar in their mathematical definitions, but incur very different challenges from the network prediction perspective.

In particular, the associative embedding space (*i*) acquires its structure through implicit supervision, (*ii*) requires more elaborate mechanisms for decoding, and (*iii*) may require the network to work in a global manner to solve IS, *i.e.*, against translation-equivariance, as discussed in Section 2.4.1. Spatial embeddings, in contrast, are pre-organized using image coordinates, which allows the network to work in a local and translation-equivariant way, as discussed in Section 2.4.2, and allows the decoding to be more straightforward. However, those advantages come at the cost of a limited ability to handle close instances.

In Section 5.1, we present a unified end-to-end mathematical framework for IS that supports spatial and associative embeddings and helps mixing them. Sections 5.2 to 5.4 introduce components to support the convergence of associative embeddings, which we assess in Section 5.5. While seemingly counterproductive, this allows us to isolate the most reliable set of com-

ponents to overcome the weakness of spatial embeddings to handle close instances. We introduce hybrid embedding methods in Section 5.6 and assess them on real and synthetic data in Section 5.7. Section 5.8 discusses how hybrid embeddings correspond to reorganizing the predictions so as to comply with the constraints of the decoding, and how this idea could transfer to a variety of tasks. Section 5.9 concludes.

5.1 A unified mathematical framework for embedding-based methods

Having presented different types of embedding-based methods in Section 2.4, we here extract different possibilities for their components that will be useful in the rest of this dissertation. We rely on the task definition introduced in Section 2.1.1. While we focus on associative embeddings that are linear, *i.e.*, $\Psi(p) \in \mathbb{R}^D$, for simplicity and straightforward compatibility with spatial embeddings, components from non-linear associative embeddings [KF18, KYL18] can be integrated seamlessly.

Foreground and annotated instance masks. Instance embeddings are used to separate pixels of the *foreground*, *i.e.*, pixels belonging to objects. As a reminder, each annotated instance i is provided with an annotated mask $\mathcal{I}_i^*$. We define the ground-truth foreground as $\mathcal{F}^* = \bigcup_{i=1}^{|\mathcal{I}^*|} \mathcal{I}_i^*$ and ground-truth *background* as $\mathcal{B}^* = P \setminus \mathcal{F}^*$. At inference, an estimated foreground $\mathcal{F}$ is obtained by semantic segmentation, regrouping pixels predicted to belong to *things* classes.

Network output. Let us first discuss the raw network outputs $\Phi(p)$ dedicated to the instance embeddings. We assume that these network outputs are linear, *i.e.*, that no non-linear activation function is applied. At initialization, those $\Phi(p)$ therefore usually end up with small but very noisy values around 0.

Embeddings. Associative embedding methods [NHD17] use the network outputs as pixelwise instance embeddings directly:

$$\Psi(p) = \Phi(p) , \tag{5.1}$$

which are thus also noisy around 0 at initialization. To help the embeddings to differentiate automatically across the image, pixelwise spatial embeddings are instead obtained by adding the pixel coordinates to each output [NBPG19, CCZ⁺20]:

$$\Psi(p) = \Phi(p) + p , \quad (5.2)$$

therefore differentiating pixels of remote instances automatically.

Centroid. The centroid o_i^* of instance i is the value that its pixels embeddings $\Psi(p)$ should try to reach to form a cluster. Associative embedding methods associate to each instance a centroid that is the average embedding over its pixels

$$o_i^* = \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} \Psi(p) . \quad (5.3)$$

Spatial embeddings most often rely on fixed centroids each defined as the center of mass of its instance’s mask:

$$o_i^* = \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} p . \quad (5.4)$$

It may help intuition to visualize that, so far, associative embeddings imply a constant output over each instance whose exact value has to emerge at training, *i.e.*, $\Phi(p) = o_{I^*(p)}^*$, while spatial embeddings will instead require the network to output at each pixel an offset towards the instance center of mass, $\Phi(p) = o_{I^*(p)}^* - p$. Additionally, with spatial embeddings, both associative and spatial definitions of centroids coincide with the instance center of mass when $\Phi(p) \approx 0$ (usually the case at initialization).

Pull loss. To train embeddings $\Psi(p)$ to approximate centroids $o_{I^*(p)}^*$, both associative and spatial embeddings can rely on the same kinds of pull losses to pull instance embeddings together, usually L1 or L2 error norms, either squared or not, *i.e.*,

$$L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \|\Psi(p) - o_{I^*(p)}^*\|_a^b , \quad (5.5)$$

with $a \in \{1, 2\}$ and $b \in \{1, 2\}$ being hyperparameters. Optionally, the loss may be relaxed inside a given margin, such as in [DBNVG17]:

$$L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \left| \|\Psi(p) - o_{I^*(p)}^*\|_a^b - M_{\text{pull}}^b \right|_+^c. \quad (5.6)$$

The combination of b and c helps modulating the gradient. Pull losses can also act between pixels, such as in Equation (2.13).

Push loss. The push loss is commonly used when the centroid is learnable, *e.g.*, when Equation (5.3) is used, in order to obtain distinct centroids. Push losses may rely on different granularities:

- (i) pixel-pixel: between the embeddings of pairs of pixels from different instances, as in Equation (2.14) [NHD17],
- (ii) centroid-pixel: between an instance centroid and pixel embeddings from other instances, *e.g.*,

$$L_{\text{push}} = \frac{1}{|\mathcal{F}^*|} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{\substack{p \in \mathcal{I}_j^* \\ j \neq i}} |M^b - \|o_i^* - \Psi(p)\|_a^b|_+^c, \quad (5.7)$$

with $a \in \{1, 2\}$, $b \in \{1, 2\}$ and $c \in \{1, 2\}$,

- (iii) centroid-centroid: between centroids of different instances [DBNVG17], *e.g.*,

$$L_{\text{push}} = \frac{1}{|\mathcal{I}^*|(|\mathcal{I}^*| - 1)} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{j \neq i} |M^b - \|o_i^* - o_j^*\|_a^b|_+^c, \quad (5.8)$$

with the same possibilities for a , b and c . While it may seem counter-intuitive to apply a loss on ground-truth terms, when learnable, the centroid o_i^* is defined as the average embedding over instance i , and therefore depends on the network prediction.

The exact parameters and normalization factors vary in the literature. Other mechanisms than push-pull losses exist in order to optimize embeddings with learnable centroids, *e.g.*, the Lovasz-Hinge from Equation (2.21) [NBPG19]. Due to difficulties encountered to make embeddings converge with the Lovasz-Hinge loss and pixel-pixel ones, we focus our study on (ii)

and (iii). Intuition will be provided in Section 5.5 as to why some supervisions may not be equipped to make associative embeddings converge.

Explicit embedding regularization. A network with learnable centroids has moving ground-truth instance centroids, which may cause its outputs to unnecessarily reach extreme values. In order to keep those outputs bounded and to encourage the use of all embedding dimensions, regularization can be used through an explicit loss term. While the norm of embeddings or centroids can be regularized in the context of associative embeddings [DBNVG17], for uniformity and compatibility with both associative and spatial embeddings, we directly regularize the l2-norm of the network outputs $\Phi(p)$:

$$L_{\text{reg}} = \frac{1}{|P|} \sum_{p \in P} \|\Phi(p)\|_2 . \quad (5.9)$$

Decoding. From predicted embeddings, some IS methods directly use general clustering algorithms such as Mean-Shift Clustering [DBNVG17], DBSCAN or OPTICS [KGC18] to obtain predicted instance masks, which all imply significant runtime costs. In this dissertation, we instead formalize and study more runtime-efficient clustering strategies as this increases the practical and theoretical value of our work, rendering it more end-to-end and more relatable to other tasks, *e.g.*, to pose estimation and object detection.

Seeds. When targeting an efficient clustering strategy, obtaining a good set of *seeds*, *i.e.*, candidate centroids, is a crucial part of the clustering, and basically amounts to extracting a set of detections. Seeds can be obtained from the histogram of the output embeddings [NHD17], *e.g.*,

$$\mathcal{O} = \text{NMS}_{Q(\psi)} \# \{p \in \mathcal{F} : Q(\Psi(p)) = Q(\psi)\} , \quad (5.10)$$

where $Q(\cdot)$ achieves the quantization of the embeddings into bins with chosen resolution. (NMS notation is introduced in Section 2.2.5) However, those histograms become significantly harder to compute and process when embeddings are pixelwise or high-dimensional. For this reason, rather than from the histogram, efficient methods extract seeds from 2-D seed maps $\mathbf{H} \in \mathbb{R}^{H \times W}$ [NBPG19, CCZ⁺20], extracting their local maximas spatially. When the embedding space is related to the image space, *e.g.*, with spatial

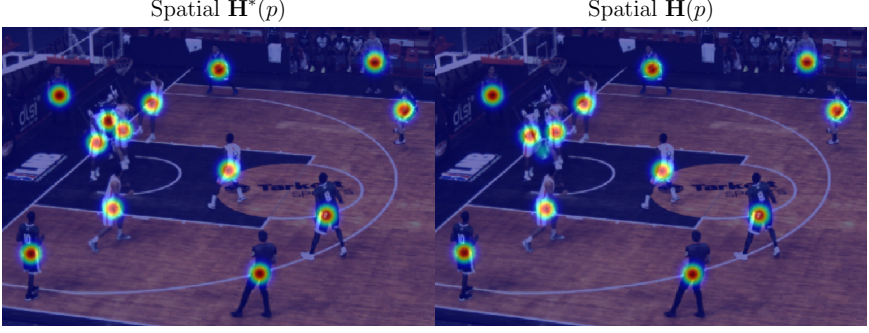


Figure 5.1: Example ground-truth map $\mathbf{H}^*$ and predicted seed map $\mathbf{H}$ obtained with Equation (5.11). In this example, an instance centroid is missed in the melee and will lead to a false negative.

embeddings, the network can be trained to output Gaussian kernels at the location of centroids, just as keypoints can be detected in images [CCZ⁺20]:

$$\begin{cases} \mathbf{H}^*(p) = \max_i \exp\{-\frac{1}{2\sigma^2}\|p - o_i^*\|_2^2\} \\ \mathbf{H}(p) \approx \mathbf{H}^*(p) \\ \mathcal{O} = \text{NMS}(\mathbf{H}, P) . \end{cases} \quad (5.11)$$

Example ground-truth map $\mathbf{H}^*$ and predicted map $\mathbf{H}$ are provided in Figure 5.1. Unlike keypoints, those centroids do not correspond to visually striking patterns. Having the network to grasp that each is the center of mass of an instance mask is already a small miracle. In practice, we observe that Convolutional Neural Networks (CNNs) estimate those maps poorly when centroids move to less characteristic positions.

When the embedding space is not related to the image space or that centroids are not at the center of mass of instance masks, it becomes necessary to resort to other mechanisms, such as quality seed maps, *i.e.*, seed maps related to the reliability of the embedding in every pixel. For instance, the network can be supervised to estimate the quality of its predicted embeddings, as proposed in [NBPG19], resulting in the prediction of what we refer to as the Learned Quality Seed Map (LQSM), *i.e.*, $\mathbf{H}(p)$ in

$$\begin{cases} \mathbf{H}^*(p) = \mathbb{1}_{\{p \in \mathcal{F}^*\}} \exp\{-\frac{1}{2\sigma^2}\|\Psi(p) - o_{I^*}^*(p)\|_2^2\} \\ \mathbf{H}(p) \approx \mathbf{H}^*(p) , \end{cases} \quad (5.12)$$

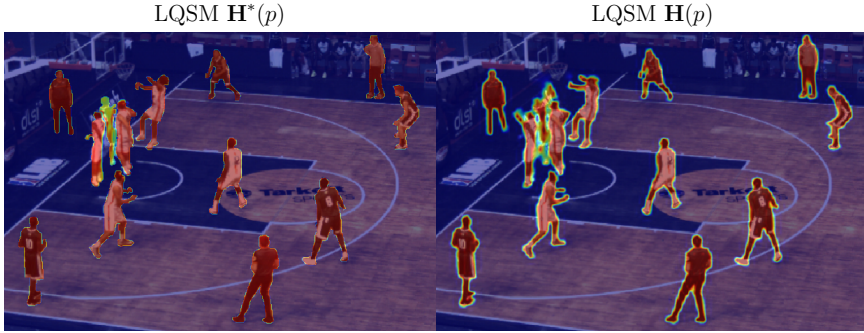


Figure 5.2: Example ground-truth and predicted seed maps obtained with Equation (5.12) on an image of the DeepSport dataset. From comparison between the two maps, it can be observed that the network is too confident about low quality embeddings in the melee, perhaps performing semantic segmentation instead of real embedding quality estimation.

that should reflect the quality of each predicted embedding. An example is provided in Figure 5.2. For convenience, we assume that predicted embeddings are unique and we introduce the source $\mathcal{P}(\psi)$ of embedding ψ as the coordinates of the pixel predicting it:

$$\mathcal{P}(\psi) = p \in P : \Psi(p) = \psi , \quad (5.13)$$

and denote its estimated quality $\mathbf{H}(\mathcal{P}(\psi))$. From a quality seed map, the set of seeds $\mathcal{O}$ can be obtained through a NMS in the embedding space:

$$\mathcal{O} = \underset{\psi \in \{\Psi(p): p \in \mathcal{F}\}}{\text{NMS}} \mathbf{H}(\mathcal{P}(\psi)) . \quad (5.14)$$

Avoiding redundancy of seeds in the embedding space is the most important component to obtain a satisfying set of seeds, as remote locations in the output may propose close seeds for the same instance. However, first performing a NMS in the spatial domain $\text{NMS}_{p \in P} \mathbf{H}(p)$ can be beneficial to cancel embeddings whose quality could be high only because of coherency, *e.g.*, at transitions such as borders between adjacent instance masks as in Figure 5.2 (right). The entire mechanism can be achieved by computing the set of pixels with maximal quality locally $\mathcal{O}^s$, and by filtering their redundant embedding predictions to obtain the final set of seeds $\mathcal{O}$:

$$\begin{cases} \mathcal{O}^s = \text{NMS}_{p \in \mathcal{F}} \mathbf{H}(p) \\ \mathcal{O} = \text{NMS}_{\psi \in \{\Psi(p) : p \in \mathcal{O}^s\}} \mathbf{H}(\mathcal{P}(\psi)) . \end{cases} \quad (5.15)$$

This two-step decoding elegantly takes into account the spatial coherency of CNN predictions and the structure expected in the embedding space, *i.e.*, the margins enforced by push and pull losses, to turn the set of predicted embeddings into a robust set of seeds.

Clustering. From the predicted seeds $\mathcal{O} = \{o_j\}$ and embeddings $\Psi(p)$, predicted instance labels $I(p)$ are most often assigned by retrieving the closest seed to each embedding, *i.e.*,

$$I(p) = \underset{j}{\operatorname{argmin}} \|o_j - \Psi(p)\|_2 . \quad (5.16)$$

When predicted instance-specific bandwidths σ_j are available, *e.g.*, to allow larger instances to have more tolerance to error [NBPG19], it is important to scale the distances accordingly:

$$I(p) = \underset{j}{\operatorname{argmin}} \frac{\|o_j - \Psi(p)\|_2}{\sigma_j} . \quad (5.17)$$

Those computations being expensive when many seeds are detected, some works prefer building instance masks greedily based on some margin M , prioritizing seeds based on their quality score $\mathbf{H}(\mathcal{P}(o_j))$ [NBPG19]. As discussed in Section 2.4.3, this leads the clustering to correspond mathematically to:

$$I(p) = \underset{j : \|o_j - \Psi(p)\|_2 < M/2}{\operatorname{argmax}} \mathbf{H}(\mathcal{P}(o_j)) . \quad (5.18)$$

For consistency in the remaining of this dissertation, and because such a hard margin M is not always available or sufficiently reliable, we rely on Equation (5.16) which still allows the whole system to process several frames per second on modern GPUs.

From $I(p)$, individual instance masks can be obtained as sets of pixels of given values

$$\mathcal{I}_j = \{p \in \mathcal{F} : I(p) = j\} . \quad (5.19)$$

Training loss. At training, the network parameters are adjusted by minimizing the global cost function L that is a weighted sum of the embedding,

seed and semantic segmentation losses:

$$\begin{cases} L_{\text{emb}} = \lambda_{\text{push}} L_{\text{push}} + \lambda_{\text{pull}} L_{\text{pull}} \\ L_{\text{seed}} = \sum_{p \in P} (\mathbf{H}(p) - \mathbf{H}^*(p))^2 \\ L_{\text{sem}} = \frac{1}{|P|} \sum_{p \in P} \text{CrossEntropy}(C(p), C^*(p)) \\ L = \lambda_{\text{sem}} L_{\text{sem}} + \lambda_{\text{seed}} L_{\text{seed}} + \lambda_{\text{emb}} L_{\text{emb}} + \lambda_{\text{reg}} L_{\text{reg}} \end{cases} \quad (5.20)$$

where each λ . is an hyperparameter. It is common to have $\lambda_{\text{push}} = \lambda_{\text{pull}}$ and to factor them in λ_{emb} .

Dimensionality of the embedding. Just like the number of dimensions can be increased for associative embeddings, to reduce the effort required to avoid collisions or to encode more features about the instance, so can be done with the offset fields output by spatial embeddings. Indeed, several offset fields can be learned and regress different coordinates that will compose higher-dimensional embeddings. Potential coordinates to consider are instances' bounding boxes [URF⁺18, YCZ⁺19], or joint locations when pose annotation is available [PZC⁺18].

Combination of embedding-based methods. The different versions of the components introduced hereabove can be combined in a wide variety of ways. Additionally, it is possible to concatenate spatial and associative embeddings, so as to benefit from the properties of both, as suggested by [NALV18] and achieved in Section 5.6.

Overview. Figure 5.3 (top) summarizes the main components at play in this chapter. Figure 5.3 (bottom) recapitulates the combinations used by the main works mentioned in this chapter. Table 5.1 recapitulates components' equations for later use. Table 5.2 regroups the mathematical definitions for embeddings and centroids that will be discussed throughout this chapter. Table 5.3 summarizes the different embedding supervisions to be encountered in the rest of this chapter.

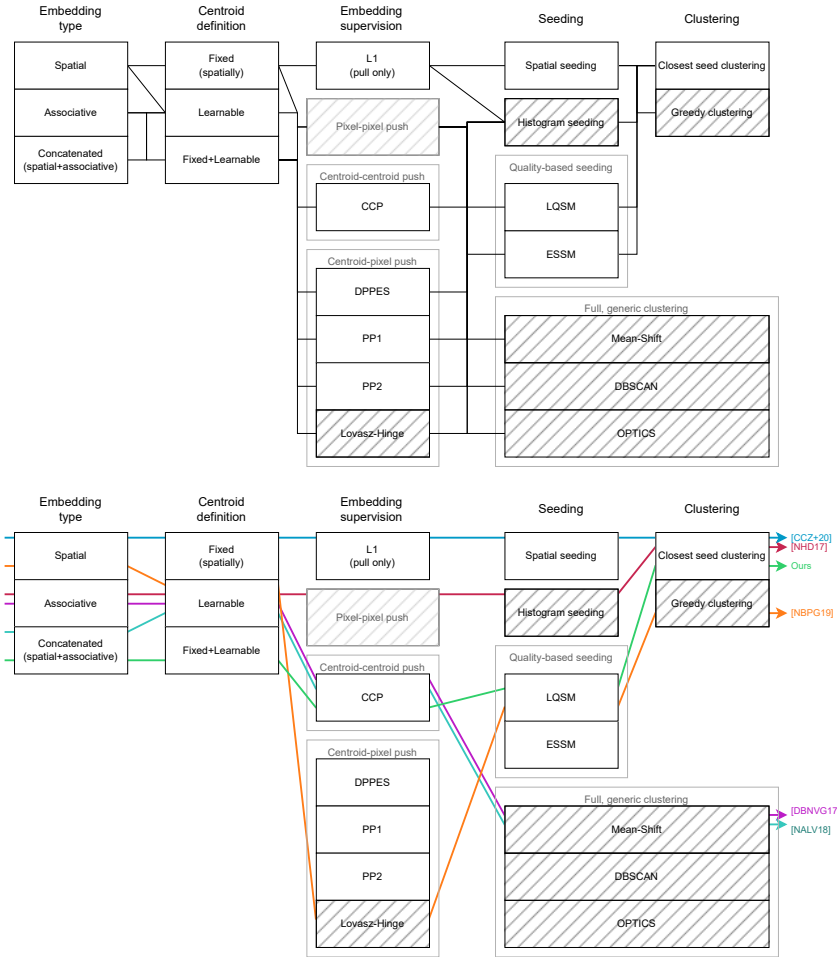


Figure 5.3: The main components identified as part of the IS framework in this chapter are aligned in columns, according to their type. Components whose block is hatched are provided for context and not part of experiments. Top: two different components being connected by a path, from left to right, can be used together. Bottom: methods can be represented each as a path through the components they use for each type. The main methods discussed in this section [NHD17, DBNMG17, NALV18, NBPG19, CCZ+20], along with the most reliable one reached at the end of this chapter, are represented in this way so as to illustrate their implementation similarities and differences. Those diagrams could be extended with more components, *e.g.*, L2 supervision, component types, *e.g.*, instance bandwidth, or methods, but are kept simple for clarity.

5.1. A unified mathematical framework for embedding-based methods

Unified framework components		
Abstract embeddings	(5.1)	$\Psi(p) = \Phi(p)$
Spatial embeddings	(5.2)	$\Psi(p) = \Phi(p) + p$
Learnable centroids	(5.3)	$o_i^* = \frac{1}{ \mathcal{I}_i^* } \sum_{p \in \mathcal{I}_i^*} \Psi(p)$
Fixed (spatial) centroids	(5.4)	$o_i^* = \frac{1}{ \mathcal{I}_i^* } \sum_{p \in \mathcal{I}_i^*} p$
Pull loss	(5.5)	$L_{\text{pull}} = \frac{1}{ \mathcal{F}^* } \sum_{p \in \mathcal{F}^*} \ \Psi(p) - o_{I^*(p)}^*\ _a^b$
Centroid-pixel push loss	(5.7)	$L_{\text{push}} = \frac{1}{ \mathcal{F}^* } \sum_{i=1}^{ \mathcal{I}^* } \sum_{\substack{p \in \mathcal{I}_i^* \\ j \neq i}} M^b - \ o_i^* - \Psi(p)\ _a^b _+^c$
Centroid-centroid push loss	(5.8)	$L_{\text{push}} = \frac{1}{ \mathcal{I}^* (\mathcal{I}^* - 1)} \sum_{i=1}^{ \mathcal{I}^* } \sum_{j \neq i} M^b - \ o_i^* - o_j^*\ _a^b _+^c$
Embedding regularization	(5.9)	$L_{\text{reg}} = \frac{1}{ P } \sum_{p \in P} \ \Phi(p)\ _2$
Histogram seeding	(5.10)	$\mathcal{O} = \text{NMS}_{Q(\psi)} \#\{p \in \mathcal{F} : Q(\Psi(p)) = Q(\psi)\}$ where $Q(\psi)$ quantizes embeddings ψ in bins
Spatial seeding	(5.11)	$\begin{cases} \mathbf{H}^*(p) = \max_i \exp\{-\frac{1}{2\sigma^2} \ p - o_i^*\ _2^2\} \\ \mathbf{H}(p) \approx \mathbf{H}^*(p) \\ \mathcal{O} = \text{NMS}(\mathbf{H}, P) \end{cases}$
Learned Quality Seed Map (LQSM)	(5.12)	$\begin{cases} \mathbf{H}^*(p) = \mathbb{1}_{\{p \in \mathcal{F}^*\}} \exp\{-\frac{1}{2\sigma^2} \ \Psi(p) - o_{I^*(p)}^*\ _2^2\} \\ \mathbf{H}(p) \approx \mathbf{H}^*(p) \end{cases}$
Quality-based seeding (Double NMS)	(5.15)	$\begin{cases} \mathcal{O}^s = \text{NMS}_{p \in \mathcal{F}} \mathbf{H}(p) \\ \mathcal{O} = \text{NMS}_{\psi \in \{\Psi(p) : p \in \mathcal{O}^s\}} \mathbf{H}(\mathcal{P}(\psi)) \end{cases}$ where $\mathcal{P}(\psi) = p \in P : \Psi(p) = \psi$
Closest-seed clustering	(5.16)	$I(p) = \underset{j}{\text{argmin}} \ o_j - \Psi(p)\ _2$
Training objective	(5.20)	$\begin{cases} L_{\text{emb}} = \lambda_{\text{push}} L_{\text{push}} + \lambda_{\text{pull}} L_{\text{pull}} \\ L_{\text{seed}} = \sum_{p \in P} (\mathbf{H}(p) - \mathbf{H}^*(p))^2 \\ L_{\text{sem}} = \frac{1}{ P } \sum_{p \in P} \text{CrossEntropy}(C(p), C^*(p)) \\ L = \lambda_{\text{sem}} L_{\text{sem}} + \lambda_{\text{seed}} L_{\text{seed}} + \lambda_{\text{emb}} L_{\text{emb}} + \lambda_{\text{reg}} L_{\text{reg}} \end{cases}$
Custom components		
Embedding Steadiness Seed Map (ESSM)	(5.21)	$\mathbf{H}(p) = \begin{cases} 1 - \frac{1}{ \mathcal{W}(p) } \sum_{p' \in \mathcal{W}(p)} \ \Psi(p') - \Psi(p)\ _2 & \text{if } p \in \mathcal{F} \\ -\infty & \text{otherwise} \end{cases}$

Table 5.1: Constitutive components of the mathematical framework unifying spatial and associative embeddings. For reuse in this dissertation.

Associative embeddings (A)

$$\begin{cases} \Psi(p) = \Phi(p) \\ o_i^* = \frac{1}{|I_i^*|} \sum_{p \in I_i^*} \Psi(p) \end{cases}$$

Spatial embeddings (S)

$$\begin{cases} \Psi(p) = \Phi(p) + p \\ o_i^* = \frac{1}{|I_i^*|} \sum_{p \in I_i^*} p \end{cases}$$

Spatial embeddings with Learnable centroids (LS)

$$\begin{cases} \Psi(p) = \Phi(p) + p \\ o_i^* = \frac{1}{|I_i^*|} \sum_{p \in I_i^*} \Psi(p) \end{cases}$$

Concatenated embeddings (C)

$$\begin{cases} \Psi(p) = [\Phi_{\{1,2\}}(p) + p \mid \Phi_{\{3,\dots\}}(p)] \\ o_{i,S}^* = \frac{1}{|I_i^*|} \sum_{p \in I_i^*} p \\ o_{i,A}^* = \frac{1}{|I_i^*|} \sum_{p \in I_i^*} \Psi_{\{3,\dots\}}(p) \\ o_i^* = [o_{i,S}^* | o_{i,A}^*] \end{cases}$$

Table 5.2: The different embedding definitions explored in this work. We restrict to linear associative embeddings for simplicity and maximum interoperability with spatial embeddings.

PP1 (Centroid-pixel push, for A & LS)

$$\begin{cases} L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \|\Psi(p) - o_{I^*(p)}^*\|_2^2 \\ L_{\text{push}} = \frac{1}{|\mathcal{F}^*|} \sum_{i=1}^{|I^*|} \sum_{\substack{p \in I_i^* \\ j \neq i}} \|M^2 - \|o_i^* - \Psi(p)\|_2\|_+ \\ L_{\text{emb}} = L_{\text{push}} + L_{\text{pull}} \end{cases}$$

PP2 (Centroid-pixel push, for A & LS)

$$\begin{cases} L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \|\Psi(p) - o_{I^*(p)}^*\|_2^2 \\ L_{\text{push}} = \frac{1}{|\mathcal{F}^*|} \sum_{i=1}^{|I^*|} \sum_{\substack{p \in I_i^* \\ j \neq i}} \|M - \|\Psi(p) - o_i^*\|_2\|_+^2 \\ L_{\text{emb}} = L_{\text{push}} + L_{\text{pull}} \end{cases}$$

Centroid-Centroid Push (for A & LS)

$$\begin{cases} L_{\text{pull}} = \frac{1}{|I^*|} \sum_{i=1}^{|I^*|} \frac{1}{|I_i^*|} \sum_{p \in I_i^*} \|\Psi(p) - o_i^*\|_2^2 \\ L_{\text{push}} = \frac{1}{|I^*|(|I^*| - 1)} \sum_{i=1}^{|I^*|} \sum_{j \neq i} \|M - \|o_i^* - o_j^*\|_2\|_+^2 \\ L_{\text{emb}} = L_{\text{pull}} + L_{\text{push}} \end{cases}$$

Concatenated Centroid-Centroid Push (for C)

$$\begin{cases} L_{\text{pull}} = \frac{1}{|I^*|} \sum_{i=1}^{|I^*|} \frac{1}{|I_i^*|} \sum_{p \in I_i^*} \|\Psi(p) - o_i^*\|_2^2 \\ M_A = \sqrt{M^2 - \|o_{i,S}^* - o_{j,S}^*\|_2^2} \\ L_{\text{push}} = \frac{1}{|I^*|(|I^*| - 1)} \sum_{i=1}^{|I^*|} \sum_{j \neq i} \|M_A - \|o_{i,A}^* - o_{j,A}^*\|_2\|_+^2 \\ L_{\text{emb}} = L_{\text{pull}} + L_{\text{push}} \end{cases}$$

DPPES (Centroid-pixel push, for A & LS)

$$\begin{cases} L_{\text{push}} = \sum_p \sum_{j=1}^{|I^*|} w_{\text{push}}^\times(p, j) D_{\text{push}}(p, j) \\ D_{\text{push}}(p, j) = \|\Psi(p) - (\Psi(p) + o_{I^*(p)}^* - o_j^*)^\times\|_2 \\ w_{\text{push}}(p, j) = \begin{cases} 1 & \text{if } I^*(p) \neq j \text{ and } \|\Psi(p) - o_j^*\| < \beta \\ 0 & \text{otherwise} \end{cases} \\ L_{\text{pull}} = \sum_p w_{\text{pull}}^\times(p) D_{\text{pull}}(p) \\ D_{\text{pull}}(p) = \|\Psi(p) - o_{I(p)}^*\|_2 \\ w_{\text{pull}}(p) = 1 \\ L_{\text{emb}} = \frac{L_{\text{pull}} + L_{\text{push}}}{\sum_p w_{\text{pull}}^\times(p) + \sum_{p,j} w_{\text{push}}^\times(p, j)} \end{cases}$$

Concatenated DPPES (Centroid-pixel push for C)

$$\begin{cases} L_{\text{push}} = \sum_p \sum_{j=1}^{|I^*|} w_{\text{push}}^\times(p, j) D_{\text{push}}(p, j) \\ D_{\text{push}}(p, j) = \left\| \left(\Psi(p) - (\Psi(p) + o_{I^*(p)}^* - o_j^*)^\times \right)_{\{3,\dots\}} \right\|_2 \\ w_{\text{push}}(p, j) = \begin{cases} 1 & \text{if } I^*(p) \neq j \text{ and } \|\Psi(p) - o_j^*\| < \beta \\ 0 & \text{otherwise} \end{cases} \\ L_{\text{pull}} = \sum_p w_{\text{pull1}}^\times(p) D_{\text{pull1}}(p) + w_{\text{pull2}}^\times(p) D_{\text{pull2}}(p) \\ D_{\text{pull1}}(p) = \left\| \left(\Psi(p) - o_{I(p)}^* \right)_{\{1,2\}} \right\|_2 \\ D_{\text{pull2}}(p) = \left\| \left(\Psi(p) - o_{I(p)}^* \right)_{\{3,\dots\}} \right\|_2 \\ w_{\text{pull1}}(p) = 1 \\ w_{\text{pull2}}(p) = 1 \\ L_{\text{emb}} = \frac{L_{\text{pull}} + L_{\text{push}}}{\sum_p (w_{\text{pull1}}^\times(p) + w_{\text{pull2}}^\times(p)) + \sum_{p,j} w_{\text{push}}^\times(p, j)} \end{cases}$$

Table 5.3: Summary of the supervisions explored in this work: PP1, PP2, DPPES and CCP. Concatenated embeddings use adjusted DPPES and CCP so as to maximize the feedback of the associative part.

5.2 Embedding Steadiness Seed Map (ESSM)

As discussed in Section 5.1, a quality seed map is introduced in [NBPG19] that the network has to learn to estimate. While it can accomodate a wide variety of formulations, this mechanism may introduce one additional task that may be non-trivial for the network: estimating the quality of its predicted embedding in every pixel.

Inspired by the clustering algorithm from Chapter 4 (Algorithm 1), we propose to rely on the conjecture that predicted embeddings are constant in inner regions of instances, so that most pixels may provide reasonable seeds. In contrast, embeddings predicted for pixels at the frontier of two instances are suspended between the two centroids, implying high variability when the centroids are sufficiently separated. The validity of this conjecture is probably subject to objections, yet we propose that it could be useful. Relying on it, we introduce a learning-free quality seed map, the Embedding Steadiness Seed Map (ESSM), based on the local stability of the output embeddings. We propose, for each pixel p , to compute the *mean absolute deviation* of embeddings predicted around it with respect to its own embedding, *i.e.*, how much its embedding is representative of theirs:

$$\mathbf{H}(p) = \begin{cases} 1 - \frac{1}{|\mathcal{W}(p)|} \sum_{p' \in \mathcal{W}(p)} \|\Psi(p') - \Psi(p)\|_2 & \text{if } p \in \mathcal{F} \\ -\infty & \text{otherwise} \end{cases}, \quad (5.21)$$

where $\mathcal{W}(p)$ denotes the window around p over which we measure the variability. Having $\mathbf{H}(p) = 1$ means that the embedding is perfectly constant around p . We do not discard negative values as variability to be expected in correct instances depends on the formulation and the model and we prefer to leave the NMS threshold as only parameter. This seed map can be post-processed with the double NMS from Equation (5.15). Figure 5.4 illustrates the proposed seed map.

5.3 Centroid-pixel push

As presented in Section 5.1, push losses can act at a centroid-pixel granularity, *i.e.*, each instance repels embeddings of other instances' pixels from their own centroids. This systematic repelling could impede the convergence of embeddings when instance embedding distributions are overlapping, *e.g.*, at initialization with associative embeddings.

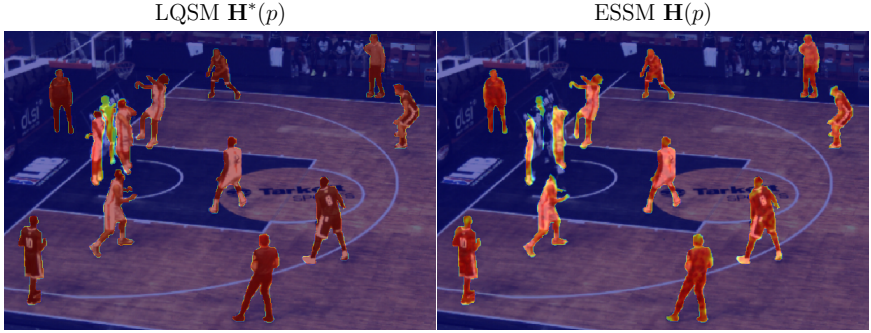


Figure 5.4: Left: Ground-truth “quality” of the embedding as defined by ground-truth LQSM in Equation (5.12). Right: Embedding Steadiness Seed Map (ESSM) from Equation (5.21), truncated in $[0, 1]$. It can be seen that the ESSM automatically avoids regions with unstable embedding, but may fail to recommend the best seeds.

Formulating a centroid-pixel push loss that avoids this problem is non-trivial for two reasons. First, it is not straightforward to craft an analytical loss that allows an embedding to cross an arbitrary number of “wrong” centroids to reach the right one. Second, a certain balance has to be struck between the push and the pull losses.

We illustrate the difficulty of crafting centroid-pixel supervisions with two push-pull baselines, PP1 and PP2, and later propose to address this issue, introducing the Direct Push-Pull Embedding Supervision (DPPES). Appendix A.3 illustrates the convergence difficulties around associative embeddings more thoroughly.

5.3.1 Baselines

To illustrate the impact of embedding convergence barriers, we introduce two potential push-pull supervisions derived from Equations (5.5) and (5.7). The first, we name PP1, is analogous to that used for team discrimination

in Chapter 4:

$$\begin{cases} L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \|\Psi(p) - o_{I^*(p)}^*\|_2^2 \\ L_{\text{push}} = \frac{1}{|\mathcal{F}^*|} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{\substack{p \in \mathcal{I}_j^* \\ j \neq i}} |M^2 - \|o_i^* - \Psi(p)\|_2^2|_+ \\ L_{\text{emb}} = L_{\text{push}} + L_{\text{pull}} ; \end{cases} \quad (5.22)$$

while the second, PP2, enforces the margin in a more conventional way, *i.e.*, decreasing as the margin gets closer to being satisfied:

$$\begin{cases} L_{\text{pull}} = \frac{1}{|\mathcal{F}^*|} \sum_{p \in \mathcal{F}^*} \|\Psi(p) - o_{I^*(p)}^*\|_2^2 \\ L_{\text{push}} = \frac{1}{|\mathcal{F}^*|} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{\substack{p \in \mathcal{I}_j^* \\ j \neq i}} |M - \|\Psi(p) - o_i^*\|_2|_+^2 \\ L_{\text{emb}} = L_{\text{push}} + L_{\text{pull}} . \end{cases} \quad (5.23)$$

Push and pull forces induced by PP1 and PP2 are illustrated in Figure 5.5. It can be observed that the two forces often have antagonist goals when instance centroids are collapsed, and therefore delay embedding convergence. As we show in Appendix A.3, the convergence barriers in the optimization process are more important when more instances or more points get involved, and when embedding space is lower-dimensional.

5.3.2 Direct Push-Pull Embedding Supervision (DPPES)

To overcome barriers in embedding convergence during training, we allow embeddings that are separated from their instance centroid by wrong centroids to traverse the latter's region in a way that they eventually leave their influence. Considering a pixel p , its centroid $\overrightarrow{o_i^*}$ and another close centroid o_j^* , we always guide $\Psi(p)$ in the direction $\overrightarrow{o_j^* o_i^*}$, *i.e.*, always leading both centroids to differentiate more, as o_i^* is computed from $\Psi(p)$. In order to stick to Panoptic-Deeplab's unit-magnitude gradient, we formulate our push loss as the product of a function $D_{\text{push}}(p, j)$ with unit-magnitude gradient, only specifying the gradient direction of the loss with respect to the network output $\Psi(p)$, with a binary weight $w_{\text{push}}(p, j)$ cancelling the contributions

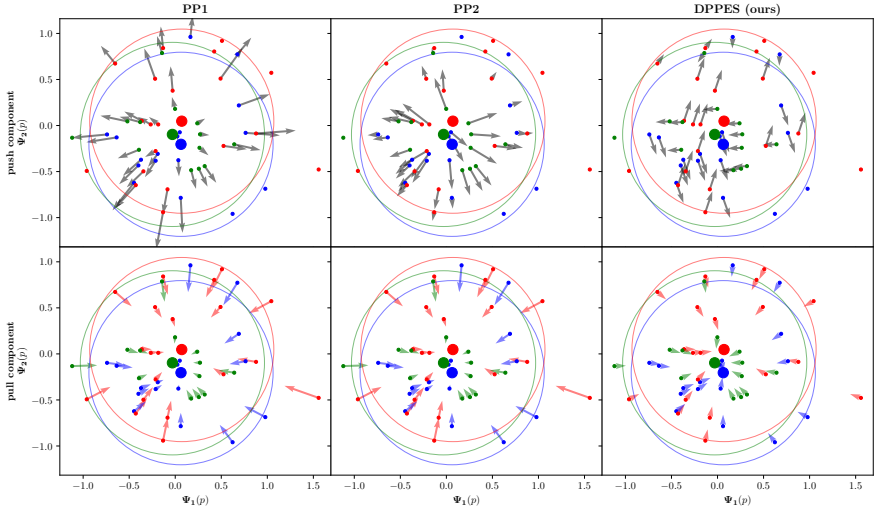


Figure 5.5: Illustration of the impact of optimizing the push and pull components of PP1, PP2 and DPPES objectives on 2-D embeddings at initialization, given an enforced margin $M = 1$. This illustration lies in the embedding space. Each Small dot represents an output embedding and is colored according to its instance. Bigger dots represent the instance centroids, *i.e.*, the center of mass of output embeddings with the same colors. Arrows indicate the direction and magnitude (proportionally) of the optimization step performed on each embedding considering a given component of the loss. Upper row shows how push components of PP1 and PP2 repel embeddings from their centroids when those are collapsed (always the case at initialization of associative embeddings), impeding embedding convergence. DPPES avoids such convergence barriers and performs updates with more uniform magnitudes. Convergence simulations are performed and discussed in Appendix A.3.

of correct pixels. The unit-magnitude gradient of the $D_{\text{push}}(p, j)$ function is most easily obtained by truncating the gradient of some of its terms, as suggested in Appendix A.2 where we introduce Gradient-Programming. The resulting push loss L_{push} then amounts to a sum over pixels and annotated instances:

$$\left\{ \begin{array}{l} L_{\text{push}} = \sum_p \sum_{j=1}^{|\mathcal{I}^*|} w_{\text{push}}^\times(p, j) D_{\text{push}}(p, j) \\ D_{\text{push}}(p, j) = \|\Psi(p) - (\Psi(p) + o_{I^*(p)}^* - o_j^*)^\times\|_2 \\ w_{\text{push}}(p, j) = \begin{cases} 1 & \text{if } I^*(p) \neq j \text{ and } \|\Psi(p) - o_j^*\| < \beta \\ 0 & \text{otherwise} \end{cases} \end{array} \right. , \quad (5.24)$$

where “ $\times$ ” denotes gradient truncation, meaning that $(\Psi(p) + o_{I^*(p)}^* - o_j^*)^\times$ is considered as a constant for optimization: its only impact is to help define the direction in which the embedding should be updated by backpropagation. Figure 5.5 (top right) illustrates the resulting optimization steps. In addition to this push loss, a simple L2 loss can serve as pull loss, which we decompose into $D_{\text{pull}}(p)$ and $w_{\text{pull}}(p)$ for later reference:

$$\left\{ \begin{array}{l} L_{\text{pull}} = \sum_p w_{\text{pull}}^\times(p) D_{\text{pull}}(p) \\ D_{\text{pull}}(p) = \|\Psi(p) - o_{I(p)}^*\|_2 \\ w_{\text{pull}}(p) = 1 \end{array} \right. . \quad (5.25)$$

Both losses are summed in L_{emb} , which we balance with respect to the other losses, *i.e.*, L_{sem} and L_{seed} , thanks to normalization with respect to the gradients weights:

$$L_{\text{emb}} = \frac{L_{\text{pull}} + L_{\text{push}}}{\sum_p w_{\text{pull}}^\times(p) + \sum_{p,j} w_{\text{push}}^\times(p, j)} . \quad (5.26)$$

The idea is to keep a similar order of the magnitude for the gradient as the L1 loss in Panoptic-Deeplab (in Equation (2.18)) to prevent the supervision balance from changing too heavily.

5.4 Centroid-Centroid Push (CCP)

In contrast to the other associative embedding methods, it is proposed in [DBNKG17] to push centroids away from each other, as in Equation (5.8). We set the pull margin to zero to achieve near-constant embedding required for our clustering and define L_{emb} according to their formulation:

$$\begin{cases} L_{\text{pull}} = \frac{1}{|\mathcal{I}^*|} \sum_{i=1}^{|\mathcal{I}^*|} \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} \|\Psi(p) - o_i^*\|_2^2 \\ L_{\text{push}} = \frac{1}{|\mathcal{I}^*|(|\mathcal{I}^*| - 1)} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{j \neq i}^{|\mathcal{I}^*|} |M - \|o_i^* - o_j^*\|_2|_+^2 \\ L_{\text{emb}} = L_{\text{pull}} + L_{\text{push}} . \end{cases} \quad (5.27)$$

The quadratic terms make this loss very similar to 5.23, except that it pushes centroids away from each other instead of individual embeddings away from centroids. It also bears some resemblance with DPPES since the push loss effectively pushes embeddings of instance i under the influence of instance j in the direction $\overrightarrow{o_j^* o_i^*}$, with the notable differences that this push loss acts uniformly over each instance's embeddings, given centroid definition, and vanishes as the centroid-centroid margin approaches satisfaction.

5.5 Associative embeddings performance

Because of the random initialization of neural network weights, associative embeddings are collapsed at the beginning of the training procedure, *i.e.*, the embedding distributions of the different instances are intertwined and the centroids lie at their centers, similarly to Figure 5.5. The structure of the embedding space has to emerge through the supervision of the pull and push losses, which may be particularly delicate. To demonstrate it, we train associative embeddings with PP1, PP2, DPPES and Centroid-Centroid Push (CCP).

We consider three different configurations for associative embeddings:

- (A1) 1-D Associative embeddings,
- (A2) 2-D Associative embeddings,
- (A5) 5-D Associative embeddings,

and compare them against

(S2) 2-D Spatial embeddings (Panoptic-Deeplab).

5-D embeddings were chosen arbitrarily in order to observe the impact of higher-dimensional embeddings while limiting memory consumption.

To compare the performance of those different methods, we train them to solve IS on the training set of the DeepSport dataset introduced in Section 2.1.2.

Training procedure. All of the experiments in this chapter are based on the public re-implementation of Panoptic-Deeplab by its authors [CCZ⁺20] augmented with ModulOM [Ist21] from Chapter 3. Each experiment consists of a 450 epochs training on an Nvidia V100 32GB GPU with batch size of 8 on the training set. We use the Panoptic-Deeplab configuration based on the HRNet-W48 backbone [WSC⁺19] pretrained on ImageNet. We set the learning rate to 10^{-2} to adapt to the lower batch size and single GPU. On every iteration, input images are resized to 1536×768 pixels with slight scale and color jitter data augmentation. The desired inter-centroid margin M in the embedding space is set to 32, σ is set to 8 accordingly. In addition to embeddings and semantic segmentation, each model is also trained to estimate the Learned Quality Seed Map from Equation (5.12) [NBPG19], except for S2 which is trained with Gaussian seed map from Equation (5.11). Loss weights $\lambda_{\text{sem}} = 1$, $\lambda_{\text{emb}} = 0.01$, $\lambda_{\text{seed}} = 200$ were kept from Panoptic-Deeplab, λ_{reg} has been found empirically to help CCP with a value of 10^{-3} , but to be detrimental to other methods (values on a logarithmic scale from 10^{-5} to 10^{-2} have been evaluated on the validation set).

Decoding. The double NMS from Equation (5.15) is used with $\mathcal{N}(p)$ being a 3×3 window centered on p and $\mathcal{N}(\psi) = \{\psi' : \|\psi' - \psi\|_2 < \frac{2M}{3}\}$. Results with both ESSM (with $\mathcal{W}(p)$ being a 5×5 window around p and NMS threshold $t = 0.1$) and the Learned Quality Seed Map (with NMS threshold $t = 0.8$) are computed. All the other parameters are kept from Panoptic-Deeplab.

Results. The PQ metric and its constituents, computed over the test set, are reported for each setup in Table 5.4. Visual examples are provided in Figures 5.6 and 5.7. Both PP1 and PP2 fail to shape the embedding space so as to achieve meaningful results: PP1 does not separate instance

Emb.	Loss	ESSM			LQSM		
		PQ	SQ	RQ	PQ	SQ	RQ
A1	PP1	03.28	69.15	04.74	00.18	78.31	00.24
	PP2	04.10	60.46	06.78	00.00	00.00	00.00
	DPPES	54.71	76.30	71.71	56.47	76.64	73.68
	CCP	76.22	84.17	90.55	78.82	84.39	93.40
A2	PP1	11.48	73.84	15.54	14.50	73.23	19.80
	PP2	04.34	63.54	06.83	00.00	00.00	00.00
	DPPES	68.93	81.37	84.71	67.22	80.40	83.61
	CCP	80.93	85.57	94.58	81.54	85.77	95.06
A5	PP1	18.28	73.61	24.84	10.12	77.19	13.11
	PP2	03.10	57.03	05.43	00.00	00.00	00.00
	DPPES	75.27	82.76	90.94	73.55	82.46	89.21
	CCP	80.87	85.11	95.01	81.33	85.07	95.61
S2	L1	83.16	86.62	96.00	83.16	86.62	96.00

Table 5.4: Results from training associative embeddings on DeepSport dataset. A1, A2 and A5 correspond to 1-D, 2-D and 5-D associative embeddings, respectively. Each is trained with PP1, PP2, DPPES and CCP supervisions introduced in Sections 5.3 and 5.4 and summarized in Table 5.3. At test-time, all models are evaluated with seeds extracted from ESSM, *i.e.*, Equation (5.21), and the Learned Seed Map, *i.e.*, Equation (5.12). Results on Spatial Embeddings (S2) are reported for comparison. It can be observed that associative embeddings perform best with 2 dimensions, CCP and LQSM, but still underperform compared to spatial embeddings.

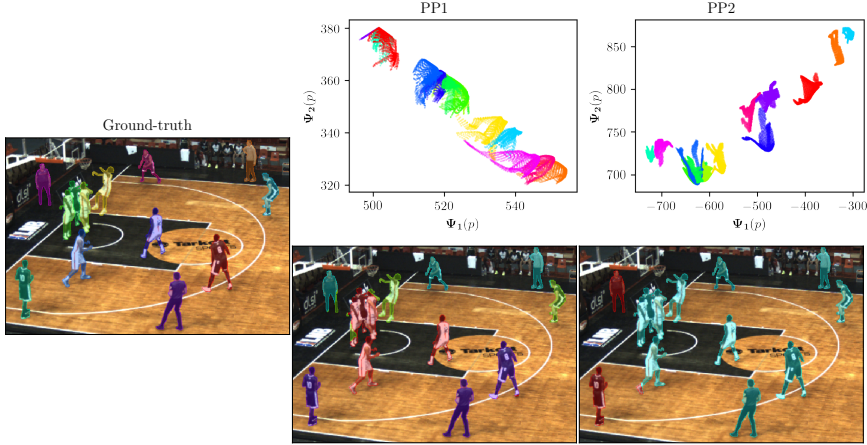


Figure 5.6: Left: ground-truth instance masks. Top: 2-D associative embedding spaces obtained after training the network using PP1 and PP2 embedding losses, embeddings are shown in the same color as their ground-truth instances on the left plot. Bottom: predicted instance labels, using new colors.

clusters sufficiently, and PP2 does not lead to sufficiently compact clusters. Both failures are the result of an inadequate balance between push and pull losses. DPPES solves that issue, but it can be seen that CCP performs better than DPPES in all cases. While having only one associative channel seems to be limiting, having two or five provides consistent results for CCP, while DPPES converges better with more channels. The proposed ESSM seed map is competitive with LQSM most of the time. Despite all our efforts, none of the associative settings outperform the spatial embeddings baseline S2, supporting the shift observed in the literature and the intuition that translation-equivariance is a strong advantage in favor of spatial embeddings, as discussed in the introduction of this chapter.

Yet, to be able to make associative embeddings converge and to decode them in such a convincing manner is pleasantly surprising, and gives us confidence that they may be able to complement spatial embeddings inside a robust framework.

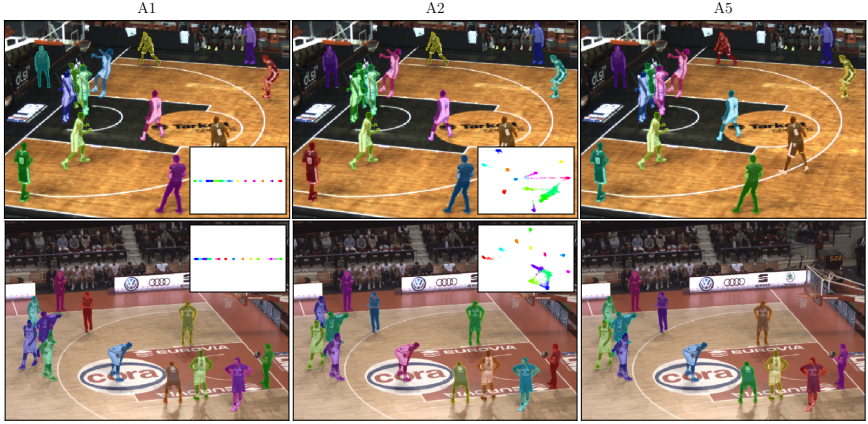


Figure 5.7: Masks predicted by 1-D, 2-D and 5-D associative embedding method, trained with the CCP supervision. Colors are randomly assigned. 1-D and 2-D embedding spaces are overlayed at the bottom-right of each image, using the same color assignment. It can be observed that instances with adjacent masks are assigned neighboring clusters embedding space with continuums between them.

5.6 Hybrid embeddings

There are many ways to mix concepts from associative and spatial embeddings in order to cope with their limitations. We choose to combine them in two ways: *(i)* spatial embeddings with learnable centroids, analogous to the method described by [NBPG19] presented in Section 2.4.3, but with fixed instance bandwidth so as to avoid unnecessary complexity, and *(ii)* fixed spatial embeddings concatenated with associative embeddings.

With those mechanisms, we aim to achieve an embedding space where close centroids are able to move away from each other in the embedding space when the fixed spatial embeddings do not sufficiently distinguish them. Obtaining distinct centroids is expected to help decoding as they are less likely to be merged by the NMS. This may also help supervising the network to better distinguish close instances, as the supervision towards the different centroids may otherwise be very ambiguous.

As a reminder, the different embedding definitions and their supervisions are respectively summarized in Tables 5.2 and 5.3.

5.6.1 Spatial embeddings with learnable centroids

In theory, spatial embeddings can become learnable by the simple use of learned centroids from Equation (5.3) and a push loss to prevent instance centroids from collapsing, *e.g.*, DPPEs in Equation (5.24). However, because the centroids lose their semantic meaning of “center of mass”, a quality seed map has to be used for decoding, *e.g.*, Equation (5.12) or Equation (5.21). Compared to spatial embeddings with fixed centroids, those components give the centroids the opportunity to reorganize in the image space.

5.6.2 Concatenated embeddings

Another hybrid variant we call “concatenated embeddings” is achieved by concatenating 2 channels of spatial embeddings with associative embedding channels. This results in a part of the centroid being fixed at the spatial center of mass of the instance mask, $o_{i,S}^*$, and the other, $o_{i,A}^*$, being learnt associatively:

$$\left\{ \begin{array}{l} \Psi(p) = [\Phi_{\{1,2\}}(p) + p \mid \Phi_{\{3,\dots\}}(p)] \\ o_{i,S}^* = \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} p \\ o_{i,A}^* = \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} \Psi_{\{3,\dots\}}(p) \\ o_i^* = [o_{i,S}^* \mid o_{i,A}^*] \end{array} \right. \quad (5.28)$$

This preserves the purpose of the spatial output to point to the center of mass of the instance, while the associative output gets the sole responsibility of disambiguating embeddings that are otherwise too close. The local aspect of the associative task renders a translation-equivariant solution suitable.

As the associative part of the embedding is the only one to be learnable, *i.e.*, to depend on Φ , we prefer to adapt DPPEs and CCP. Indeed, optimizing the norm of the concatenated vectors may fail to act on the associative part of the embedding. We illustrate this in Figure 5.8 (left).

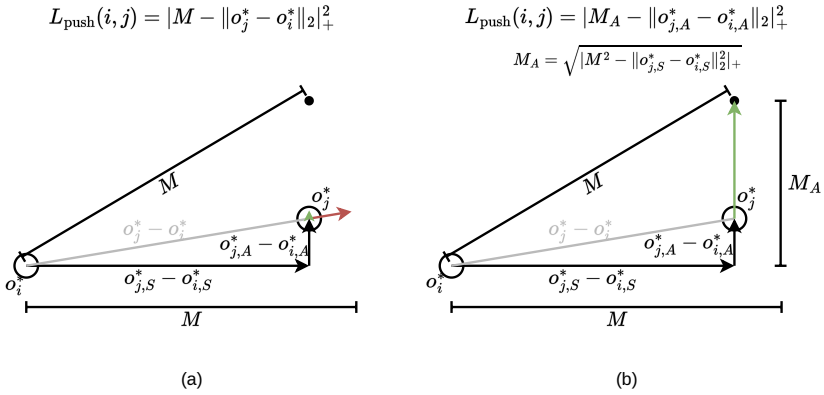


Figure 5.8: Geometric intuition behind the need for adjustments in the case of CCP with concatenated embeddings (with fixed spatial embeddings). We consider the term in the push loss that repels centroid o_j^* from o_i^* . Horizontal axis is parallel to the difference between their spatial embeddings, vertical axis is parallel to the difference between their associative embeddings. (a) without adjustment, the estimated update (arrow in red) to satisfy margin M may have underestimated magnitude and fail to act on the associative embedding. As the loss is quadratic, the gradient with respect to o_i^* and o_j^* is proportional to that estimated update, and the actual gradient to the associative part of the embedding is proportional to the length of the small green arrow. (b) with adjusted push loss, the estimated update has correct magnitude (based on M_A) and acts fully on the associative embedding. The case of DPPES is only slightly different due to the fixed gradient magnitude and the centroid-pixel nature. Thanks to the isotropy of $\|\cdot\|_2$, this reasoning is valid regardless of embedding dimensionality.

DPPEs. we therefore focus the direction of the loss on the learnable part of the embedding from Equation (5.24):

$$\left\{ \begin{array}{l} L_{\text{push}} = \sum_p \sum_{j=1}^{|\mathcal{I}^*|} w_{\text{push}}^\times(p, j) D_{\text{push}}(p, j) \\ D_{\text{push}}(p, j) = \left\| \left(\Psi(p) - (\Psi(p) + o_{I^*(p)}^* - o_j^*)^\times \right)_{\{3, \dots\}} \right\|_2 \\ w_{\text{push}}(p, j) = \begin{cases} 1 & \text{if } I^*(p) \neq j \text{ and } \|\Psi(p) - o_j^*\| < \beta \\ 0 & \text{otherwise} \end{cases} \end{array} \right. \quad (5.29)$$

We also decompose the pull mechanism from Equation (5.25) to adjust the balance between the associative pull and push gradients, and between associative and spatial pull gradients:

$$\left\{ \begin{array}{l} L_{\text{pull}} = \sum_p w_{\text{pull1}}^\times(p) D_{\text{pull1}}(p) + w_{\text{pull2}}^\times(p) D_{\text{pull2}}(p) \\ D_{\text{pull1}}(p) = \left\| \left(\Psi(p) - o_{I(p)}^* \right)_{\{1, 2\}} \right\|_2 \\ D_{\text{pull2}}(p) = \left\| \left(\Psi(p) - o_{I(p)}^* \right)_{\{3, \dots\}} \right\|_2 \\ w_{\text{pull1}}(p) = 1 \\ w_{\text{pull2}}(p) = 1 \end{array} \right. , \quad (5.30)$$

with an updated normalization, derived from Equation (5.26), *i.e.*,

$$L_{\text{emb}} = \frac{L_{\text{pull}} + L_{\text{push}}}{\sum_p (w_{\text{pull1}}^\times(p) + w_{\text{pull2}}^\times(p)) + \sum_{p, j} w_{\text{push}}^\times(p, j)} . \quad (5.31)$$

CCP. For centroid-centroid push loss, we optimize directly the associative part to satisfy the margin M_A , which is complementary to $\|o_{i,S}^* - o_{j,S}^*\|_2$ for $\|o_i^* - o_j^*\|_2$ to satisfy margin M , simply modifying Equation (5.27):

$$\left\{ \begin{array}{l} L_{\text{pull}} = \frac{1}{|\mathcal{I}^*|} \sum_{i=1}^{|\mathcal{I}^*|} \frac{1}{|\mathcal{I}_i^*|} \sum_{p \in \mathcal{I}_i^*} \|\Psi(p) - o_i^*\|_2^2 \\ M_A = \sqrt{M^2 - \|o_{i,S}^* - o_{j,S}^*\|_2^2} \\ L_{\text{push}} = \frac{1}{|\mathcal{I}^*|(|\mathcal{I}^*| - 1)} \sum_{i=1}^{|\mathcal{I}^*|} \sum_{j \neq i} |M_A - \|o_{i,A}^* - o_{j,A}^*\|_2|_+^2 \\ L_{\text{emb}} = L_{\text{pull}} + L_{\text{push}} . \end{array} \right. \quad (5.32)$$

An illustration of the impact of the mechanism is provided in Figure 5.8 (right).

5.7 Hybrid embeddings performance

We complement our set of embedding topologies with the following ones:

- (SL2) 2-D Spatial embeddings with Learnable centroids,
- (C3) 3-D Concatenated embeddings (2-D fixed spatial + 1-D associative),
- (C5) 5-D Concatenated embeddings (2-D fixed spatial + 3-D associative).

We train each with both the DPPES and CCP supervisions, following the same procedures as in Section 5.5.

5.7.1 Hybrid embeddings on DeepSport data

Training procedure. The training procedure is the same as in Section 5.5. LS2 is trained with the same losses as associative embeddings, while C3 and C5 are trained with adjusted losses discussed in Section 5.6.2, *i.e.*, Equations (5.29) to (5.32).

Decoding. The decoding procedure is exactly the same as that described in Section 5.5 for associative embeddings.

Results. Results are provided in Table 5.5. The table shows that hybrid embeddings, despite using more elaborate decoding strategies, are able to perform on par or slightly better than spatial embeddings. Once again, CCP scores higher than DPPES, although the difference is less striking than in

Emb.	Loss	ESSM			LQSM		
		PQ	SQ	RQ	PQ	SQ	RQ
LS2	DPPES	82.79	85.97	96.31	82.89	85.79	96.62
	CCP	83.21	86.31	96.41	83.20	86.22	96.49
C3	DPPES	83.77	86.72	96.60	83.98	86.71	96.86
	CCP	84.26	86.70	97.19	83.99	86.70	96.87
C5	DPPES	83.64	86.64	96.54	83.42	86.58	96.35
	CCP	84.04	86.59	97.05	83.96	86.45	97.12
S2	L1	83.16	86.62	96.00	83.16	86.62	96.00

Table 5.5: Results from training hybrid embeddings on DeepSport dataset. LS2, C3 and C5 correspond to 2-D Spatial embeddings with learnable centroids, and 3-D and 5-D concatenated embeddings respectively. Each is trained with adapted DPPES and CCP supervisions introduced in Sections 5.3 and 5.4 and summarized in Table 5.3. At test-time, all models are evaluated with seeds extracted from ESSM (Equation (5.21)) and the Learned Quality Seed Map (Equation (5.12)). Results on Spatial embeddings (S2) are reported for comparison.



Figure 5.9: Predicted instance masks obtained after training the network using CCP embedding loss with the LS2, C3 and C5 hybrid embeddings introduced in Section 5.6. Embedding spaces are superimposed in 2-D and 3-D cases. For C3, the 1-D associative embedding distribution of each instance is isolated vertically. It can be seen that the 1-D associative channel only disambiguates clusters that are close in the spatial embedding.

the case of associative embeddings, see Table 5.4, due to the fact that the embedding space is partly shaped by the spatial part. ESSM and LQSM achieve very similar performance. Figure 5.9 illustrates some results with superimposed embedding spaces.

While the increase in performance of hybrid embeddings may not be striking on the PQ metric on this dataset, the models become able to deal with instances that overlapping centers of mass and to supervise erroneous pixels better. Chapter 6 focuses on partitioned instance reconnection and provides additional evidence that those are better resolved using hybrid embeddings. Table 6.3 reports concatenated embeddings to have the strongest improvement in the rate of remote pixels being reconnected to their mask.

5.7.2 Hybrid embeddings on synthetic data

While the previous section reveals that hybrid embeddings can bring some improvements on real-life data, we here provide a more striking demonstration of the ability of hybrid embeddings to distinguish overlapping instances. In common datasets, colliding centroids may correlate with other difficulties, making them complete outliers in regard to the problems to solve, *e.g.*, more border pixels that are inherently ambiguous, there is less evidence of the existence of the occluded objects, and some parts may appear visually disconnected to their instance. Therefore, the network may require more data to learn to deal with them. The metrics also risk to be dominated by the easy cases. We elaborate on those issues in Chapter 6, but focus here on illustrating that the inputs that are pathological for spatial embeddings can be solved accurately by hybrid embeddings.

Dataset. We synthesize a dataset with pairs of instances with overlapping centroids. The dataset is created by pasting MNIST digits so as to make their centroids overlap (before considering the occlusion, which modifies the centroid of the occluded instances). Each digit is pasted with a unique (random) color to make sure the network has the information required as to which pixels should go together. Each image has resolution 512×256 and contains 3×7 pairs of instances. An example image and its corresponding spatial seed map are provided in Figure 5.10. At least 50% of the pixels of each instance are guaranteed to be visible. The exact generation algorithm is described in more details in Section 6.2.2.

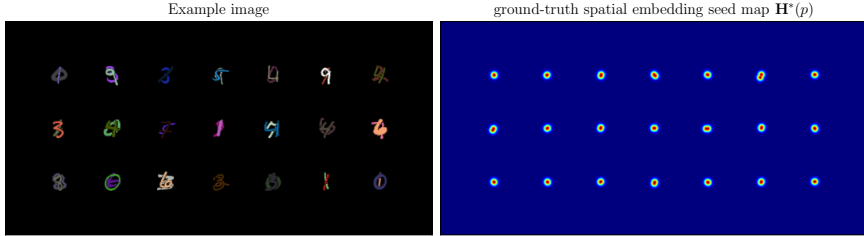


Figure 5.10: Left: example image from our synthetic dataset, obtained by pasting pairs of instances so that their centroids overlap (prior to considering occlusions). Right: the corresponding ground-truth seed map obtained. It can be seen that most Gaussian kernels still overlap significantly, constituting a challenge to decode distinct instances and assign embeddings correctly.

Training procedure. The training process is the same as in Section 5.5, except that the learning rate is set to 10^{-3} , that images are upsampled to 1536×768 , that the backbone is randomly initialized and that only 250 epochs are performed. We keep training with both DPPES and CCP.

Decoding. We use the same decoding strategy as in Section 5.5. We keep evaluating the results with both ESSM and LQSM seed maps. For the ESSM, the neighborhood $\mathcal{W}(p)$ is reduced to a 3×3 window due to the thinness of digits.

Results. Metrics are provided in Table 5.6 and illustrated examples are provided in Figure 5.11. We can observe that A2 and S2 both fail to solve this problem. LS2 and C3 can tackle it with sometimes impressive performance (97.52% and 98.43% of recognized instances respectively, using CCP+LQSM).

To our surprise, CCP requires the LQSM seed map to provide convincing results but achieves the best performance, while DPPES achieves better performance with the ESSM seed map than with LQSM. This likely has to do with how the supervision prioritizes embedding variability embedding in “inner” pixels against getting hard pixels relatively right, which are numerous due to the heavy overlap and thinness of instances. As CCP has a quadratic pull loss, it performs some kind of hard example mining and may, therefore, neglect getting correct embeddings exactly right.

Overall, C3 keeps a slight advantage over LS2, and CCP+LQSM appears

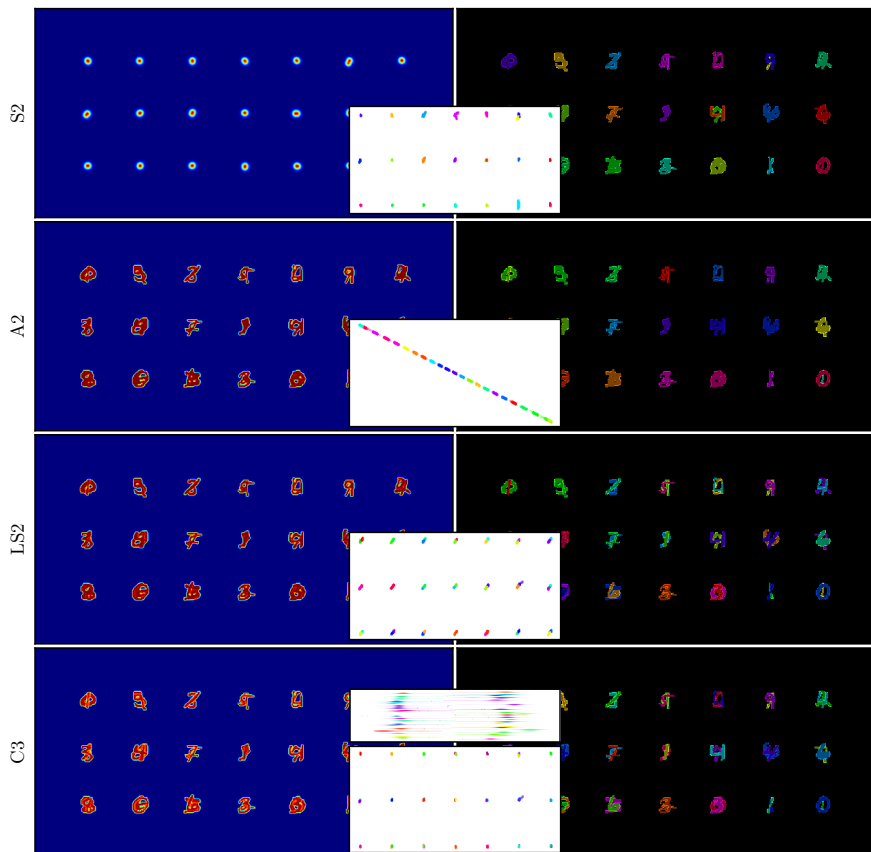


Figure 5.11: Seed maps (left) and instance masks (right) predicted by (from top to bottom) S2, A2, LS2 and C3, from input image in Figure 5.10 (left). As expected, spatial embeddings (S2) and associative embeddings (A2) struggle heavily to process individual instances, while hybrid methods are able to extract most of them successfully.

Emb.	Loss	ESSM			LQSM		
		PQ	SQ	RQ	PQ	SQ	RQ
A2	DPPES	41.41	71.30	58.08	36.71	74.21	49.47
	CCP	35.04	62.52	56.04	41.84	63.96	65.41
LS2	DPPES	71.49	79.36	90.08	63.75	78.29	81.43
	CCP	45.19	66.98	67.46	87.07	89.29	97.52
C3	DPPES	71.57	80.42	88.99	65.07	78.94	82.43
	CCP	45.54	66.93	68.05	88.63	90.04	98.43
S2	L1	44.99	66.16	68.00	44.99	66.16	68.00

Table 5.6: Results from training spatial, associative and hybrid embeddings on a synthetic dataset with overlapping digits. A2, LS2, C3 correspond respectively to 2-D Associative embeddings, 2-D Spatial embeddings with learnable centroids, and 3-D Concatenated embeddings. Each is trained with adapted DPPES and CCP supervisions introduced in Sections 5.3 and 5.4 and summarized in Table 5.3. At test-time, all models are evaluated with seeds extracted from ESSM (Equation (5.21)) and the Learned Quality Seed Map (Equation (5.12)). S2 corresponds to 2-D Spatial embeddings as implemented in Panoptic-Deeplab [CCZ⁺20].

to be the most reliable combination in such challenging scenes.

5.8 Broader implications

Spatial embeddings for IS and other offset regression models have limitations that are deeply related to CNNs’ spatial coherency and the use of NMS. The neighborhood used for NMS must be small so as to allow close but correct predictions to be decoded, while remaining large enough to cancel redundant ones, resulting in a trade-off whose optimal balance sacrifices the resolution of hardest cases.

Adding associative embeddings can help lift this trade-off to allow network outputs with otherwise similar properties to be decoded individually, avoiding to cancel each other through NMS. For example, we have seen in Section 2.3 that object detectors cancel bounding box predictions that have a large IoU with higher-score predictions. Akin to spatial embeddings, bounding-box regression is achieved through the regression of offsets that are converted into bounding box coordinates. An associative embedding may help to keep bounding boxes with similar coordinates that are in fact

related to different objects. Similar mechanisms could be applied in player tracking, pose estimation, event detection and probably many other tasks, although the exact design choices will vary.

As broad future investigations, it could be interesting to (i) evaluate the impact of double NMS in Equation (5.15) and associative embeddings on other tasks with difficulties on occlusions, *e.g.*, bounding box detection, top-down IS and short-term tracking, (ii) research similar synergies of associative embeddings and sporadic features to address long term tracking and Re-ID [KDV13, SDVA23] or, more fundamentally, to investigate (iii) classification and semantic segmentation with linear associative embeddings and (iv) the impact of non-Euclidean norms in the push-pull losses and decoding.

5.9 Conclusion

In this chapter, we introduced a unified framework for spatial and associative embeddings that helped us solve the IS problem in an end-to-end manner. We extended this framework with a versatile learning-free seed map based on the local coherency of embedding predictions, *i.e.*, ESSM. We introduced a centroid-pixel push-pull supervision *i.e.*, DPPES, that avoids convergence barriers in the formation of associative embeddings using gradient programming. We also joined features from associative and spatial embeddings in order to get two hybrid embedding formulations: spatial embeddings with learnable centroids, and concatenated (spatial + associative) embeddings. The idea is that associative and spatial embeddings have complementary advantages: the associative part of the embedding helps the spatial part to handle otherwise ambiguous centroids, while the spatial part helps the associative one to work in a local, and therefore, translation-equivariant, manner.

After confirming that associative embeddings were not able to match the performance of spatial embeddings, we observed that hybrid embeddings had the potential to outperform both. In particular, concatenated embeddings achieved higher performance than spatial embeddings in all scenarios. Regarding their implementation, we came to the conclusion that the CCP was superior to centroid-pixel push losses and that the LQSM could enable better performance in very challenging scenes while the ESSM could remain competitive in less demanding conditions, relieving the network from having to estimate its embedding quality for decoding.

Although the gain in performance was limited on basketball scenes, we illustrated the fundamental improvement in expressivity in order to segment instances in a synthetic dataset with almost completely overlapping instance centroids.

Possible IS-related investigations comprise *(i)* the assessment of hybrid embeddings on more multi-class datasets, *e.g.*, Cityscapes [COR⁺16], *(ii)* the reintroduction of learnable instance bandwidth from [NBPG19], and *(iii)* a study of the impact of each factor differentiating the CCP and DPPES supervisions.

Chapter 6

Resolving hard occlusions

For all Instance Segmentation (IS) methods, a major challenge lies in solving occlusions. Some occlusions can be (relatively) simple to be dealt with, *e.g.*, when two instances from different semantic classes overlap. Our focus in this chapter is on harder occlusions, referred to as *partitioned instances*, *i.e.*, when an instance is split by another instance of a same class. The difficulty of an IS task in such cases is to distinguish instances in regions of uniform semantic class but also reconnect the separated parts.

As discussed in Section 2.3, whilst various ideas have been investigated to better handle occlusions in top-down methods [KTT21, LM16], part of their internal mechanisms limits the ability to detect and specifically handle overlapping objects. In contrast, as demonstrated in Chapter 5, through their pixelwise embeddings, bottom-up approaches do offer such capability.

In Bottom-up methods, strong occlusions between instances require sharp embedding transitions across neighboring pixels. The problem can be mitigated when an occlusion involves stuff (as illustrated in Figure 6.3, where a car is partitioned by a pole, labeled as stuff), rather than another object. This is because the output embedding vectors can simply ignore stuff, while they have to create distinct clusters for each object in the second case. Pixelwise semantic class can also help to distinguish overlapping objects, but only when those are of different classes.

No matter how successful the current IS methods are, the analysis of their performance in presence of occlusions, especially partitioned instances, is lacking. In particular, conventional metrics do not reflect the true ability of IS methods in treating (partitions caused by) occlusions. This chapter aims to provide answers to the challenges related to heavy occlusion cases.

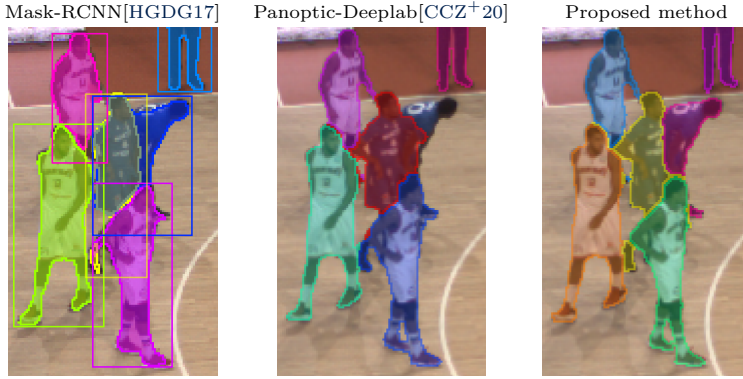


Figure 6.1: Example of IS results on a scene with strong occlusions. Both top-down (left) and bottom-up (middle) methods struggle in cases of strong occlusions, especially to reconnect partitioned instances. This chapter *(i)* introduces a new metric, named Partitioned Instance Quality (PIQ), that is appropriate to quantify those difficulties, and *(ii)* proposes dedicated data augmentation and ambiguity-driven embedding supervision (ADES) mechanisms to address them.

It focuses on the most challenging cases, involving significant occlusions within a single class of objects.

The major part of our work is based on the Panoptic-Deeplab framework, explained in Section 2.4.2 and illustrated in Figure 6.2, as it is an established end-to-end solutions with no sampling involved anywhere before the end of the network. Our goal is to provide tools and metrics to improve and understand the behavior of IS methods in presence of occlusions. Section 6.1 surveys IS methods, dataset augmentation methods and usual IS metrics from the literature. In Section 6.2,

- we introduce a new IS quality metric, referred to as Partitioned Instance Quality (PIQ), which focuses on the IS artifacts induced by occlusions. See Section 6.2.1.
- We develop a family of synthetic datasets based on copy-pasting of instances allowing us to control the degree of occlusion in the images; hence, providing insights in understanding the impact of the training data on IS performance. See Section 6.2.2.
- We propose a novel loss function leveraging online hard example mining to improve the prediction of spatial embedding vectors for occluded

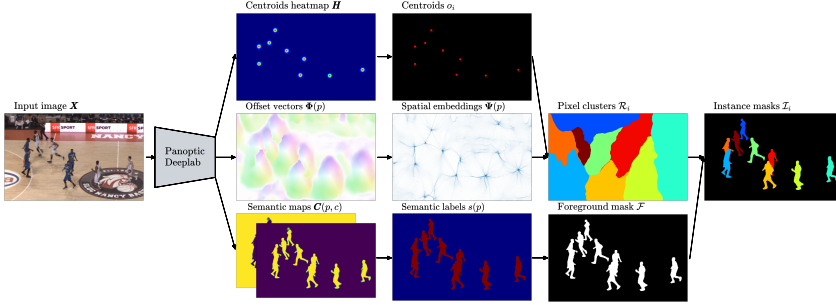


Figure 6.2: An overview of Panoptic-Deeplab [CCZ⁺20] used for IS. The network produces a heatmap of instances’ centroids, offsets vectors, and semantic segmentation maps. Local maximums are extracted from the heatmap to obtain candidate instance centroids and spatial embeddings are computed from offset vectors. Those information are fused to cluster the pixels. Instance labels are then computed from the output of the clustering and pixelwise semantic labels.

instances. In particular, we modulate the gradient of the L1 loss so that more attention is put on the update of the embeddings of the most ambiguous pixels in occluded instances. See Section 6.2.3.

We perform systematic experiments on our synthetic datasets in Section 6.3 and explore the impact of instances’ characteristics, *e.g.*, occlusion level, shape, and color content, on the performance of the network to reconnect instance partitions.

Subsequently, we demonstrate in Section 6.4 the relevance of using our dataset synthesis for data augmentation on a real dataset. Section 6.5 concludes. Our findings, *i.e.*, best illustrated in Table 6.3, suggest that heavy occlusions becoming frequent at training not only improve the ability of the network to handle occlusions by themselves, but also convey mechanisms addressing them more impact.

6.1 Related works

Bottom-up IS. Related literature over bottom-up IS has been surveyed in Section 2.4. Although most of those works show results on datasets with occlusions, they do not dedicate explicit analysis of occlusions and their contribution in classical IS quality metrics.

Dataset augmentation. Dataset augmentation techniques based on copy-pasting instances across images have been introduced in [DMS18, GCS⁺21]. Dvornik *et al.* [DMS18] modeled the visual context in order to paste instances in realistic locations. Ghiasi *et al.* [GCS⁺21] demonstrated random copy-pasting for dataset augmentation. Both of those works reported that unrealistic visual artifacts injected at training do not impede the network performance on real data. Our work leverages this observation, and demonstrates that copy-pasting operations can specifically help solve the instance partitions induced by occlusions.

Conventional IS quality metric. IS methods are conventionally benchmarked using the Panoptic-Quality (PQ) or Mean Average Precision (mAP) metrics introduced in Section 2.1.3. In this chapter, we focus on the PQ and on a metric of our own, that focuses on partitioned instances.

6.2 Proposed Methods

We here introduce *(i)* a novel metric that can quantify the performance of an IS model for partitioned instances, *(ii)* a family of datasets that allows us to investigate the impact of an instance occlusion bias during training and evaluation, and *(iii)* a training loss function focusing on hard occluded instances.

6.2.1 A metric tailored for partitioned instances

Conventional metrics, such as PQ and mAP, are not relevant to measure how well an IS method can handle occlusions. The fraction of samples affected by occlusions is limited in datasets, and conventional metrics average performance on the whole datasets. Occlusions have, therefore, only a narrow impact on such metrics, while the metrics' variability increases with the variety of classes, sizes, shapes and patterns present in the dataset, across which models may perform differently. For the metrics and models to focus as much as possible on the occlusion handling problem, staging our experiments in single-class, things-only contexts, both for training and performance evaluation is important, but not sufficient.

We now turn our attention to the definition of PIQ composed of two parts. Partitioned Instance Recall (PIR) captures the recall of the partitioned instances, while Partitioned IS Quality (PISQ) measures the ratio

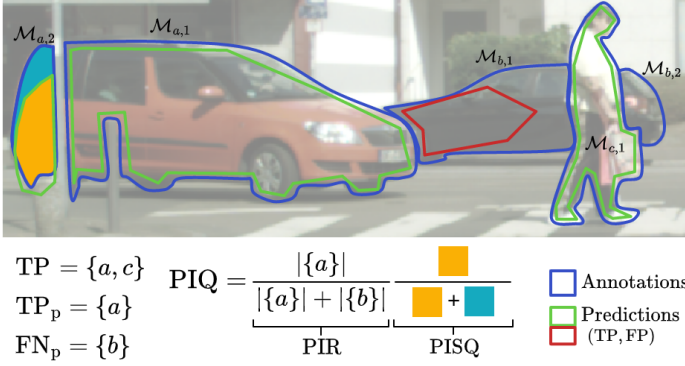


Figure 6.3: An Illustration of the PIQ computation. Instance c does not intervene, since it is not partitioned. Instance b ’s IoU with red prediction is insufficient and therefore intervenes in reducing the PIR. Instance a is partitioned and has a matched prediction; therefore, it increases the PIR and $\mathcal{M}_{a,2}$ contributes to PISQ. Predicted instances are omitted from TP and TP_p for lighter notation.

of instances’ pixels lying outside the largest partitions that are correctly reconnected to the predicted instance mask.

Let us define TP_p ⊆ TP as a TP set with respect to the partitioned instances: it contains the pairs of “annotated partitioned instances” and predicted instances whose IoU > 0.5. Similarly, FN_p ⊆ FN is the set of annotated partitioned instances with no matched predicted instance. Moreover, let $\mathcal{M}_{g,j}$ be the mask (*i.e.*, a set of pixels) of the j^{th} largest partition of annotated instance g . Using those definitions, PIQ is defined as

$$\text{PIQ} = \underbrace{\frac{|\text{TP}_p|}{|\text{TP}_p| + |\text{FN}_p|}}_{\text{PIR}} \cdot \underbrace{\frac{\sum_{(g,q) \in \text{TP}_p} \sum_{j>1} |\mathcal{M}_{g,j} \cap \mathcal{I}_q|}{\sum_{(g,q) \in \text{TP}_p} \sum_{j>1} |\mathcal{M}_{g,j}|}}_{\text{PISQ}}. \quad (6.1)$$

An example of PIQ computation is illustrated in Figure 6.3. Finally, as we may want to focus on the performance of part reconnection regardless of their semantic segmentation prediction, we propose the PISQ with oracle semantics, denoted PISQ_o, and the analogous PIQ_o = PIR · PISQ_o. To compute the PISQ_o, predicted instances masks $\mathcal{I}_q$ in Equation (6.1) can be considered as containing all the pixels for which q is the most likely predicted instance, independently from their semantic prediction. In our case, this amounts to using the ground-truth foreground mask $\mathcal{F}^*$ instead



Figure 6.4: Cropped samples from three synthetic datasets of different difficulties proposed for IS task.

of $\mathcal{F}$ in Equation (2.20) and Figure 6.2.

6.2.2 A synthetic occlusion-controlled dataset

We propose Instance Segmentation MNIST (ISMNIST), a family of datasets synthesized from MNIST dataset [LBBH98]. MNIST was developed mainly for classification task, containing only one digit per 28×28 image. Those digits are mostly white, diverse in shape when considering all classes, and in most cases, are constituted of only one connected component, *i.e.*, there is no partitioned instance. Another consideration in favor of MNIST is that digits are not plain blobs, *i.e.*, they have intricate shapes with narrow parts, and in that sense are similar to human masks and other complicated objects. The mask of every digit can be retrieved by removing the zero-intensity background from the image.

We play on three key characteristics to generate datasets with varied degrees of difficulties: instance overlap, instance color, and instance shape.

Those datasets are composed of images generated by pasting MNIST digits around arbitrary points, designated as center of clusters. Controllable parameters for each cluster include (i) cluster center location, (ii) width, (iii) maximum number of instances, (iv) minimum proportion of visible pixels, (v) minimum distance between the instance centroids, (vi) instances' color, (vii) the MNIST classes from which each instance is selected, and (viii) the maximum number of pasting tries. The parameters (i)-(v) allow us to play on the overlap of instances, while the parameters (vi) and (vii) let us control instances' color content and shapes.

During image synthesis, clusters are generated sequentially. For each

cluster, MNIST digits are sampled randomly and pasted in the background at a randomly generated position. A sampled position is rejected if it violates the constraints on the occlusion level or instances' proximity (as in parameters (iv) and (v) above). After the maximal number of pasted instances or tries (as in parameters (iii) and (viii) above) has been reached, the next cluster is considered.

Figure 6.4 illustrates example images from three datasets synthesized by the approach above. In (i) *spaced* dataset the digits are prohibited from occluding each other and have a large margin between their centers, as a way to avoid common borders. In (ii) *packed* dataset the digits are prohibited from occluding each other, but can have closer centers and may, therefore, have common borders. In (iii) *partitioned* dataset the digits can occlude each other, but overlapping centroids are prevented. In this case, a digit's centroid is the center of mass of the remaining mask after considering occlusions. Moreover, we ensure that at least 50% of each digit's pixels remain visible.

The *packed* and *partitioned* datasets above can be generalized by considering a frequency parameter $f \in [0, 1]$ controlling the probability that a cluster is generated using the setting of *partitioned* dataset. This yields the definition of *partitioned_f* dataset having f fraction of *partitioned* clusters, and $1 - f$ fraction of *packed* clusters. Based on this convention, *packed* and *partitioned* datasets amount to *partitioned₀* and *partitioned₁*, respectively. These datasets will be used in our experiments in Section 6.3.

6.2.3 Ambiguity-Driven Embedding Supervision

As discussed in Section 2.4.2, Panoptic-Deeplab having fixed centroids, it only needs a pull loss for its embeddings. The L1 loss it uses implies a constant gradient magnitude regardless of the quality of the estimated embedding. The Smooth L1 (SL1) loss is an alternative to the L1 loss formulated as

$$\text{SL1}(p) = \begin{cases} \|\Psi(p) - o_{I^*(p)}^*\|_2^2 / 2\beta, & \|\Psi(p) - o_{I^*(p)}^*\|_2 < \beta \\ \|\Psi(p) - o_{I^*(p)}^*\|_1 - \frac{\beta}{2} & \text{otherwise,} \end{cases} \quad (6.2)$$

where $\beta > 0$ is a parameter. In our experiment we set $\beta = 4$, just smaller than half the smallest distance between two centroids in the dataset. The SL1 loss attenuates the loss for pixels whose embeddings lie closer to the

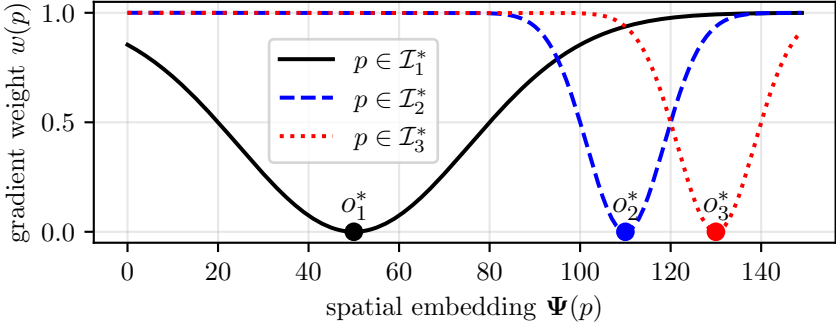


Figure 6.5: A 1-D example of the proposed gradient weights $w(p)$ with three instances. The goal is to achieve online hard example mining with fixed spatial embeddings. To this end, each embedding gradient weight is computed according to the proximity of the embedding to its centroid, relative to the available instance width in embedding space.

correct instance centroid, but does not take its actual difficulty into account.

In order to provide more supervision to more difficult instances, we introduce the Ambiguity-Driven Embedding Supervision (ADES) based on gradient programming and gradient truncation “ $\times$ ” introduced in Appendix A.2,

$$\text{ADES} = \frac{1}{\sum_{p \in P} w^\times(p)} \sum_{p \in P} w^\times(p) \|\Psi(p) - o_{I^*(p)}^*\|_1. \quad (6.3)$$

ADES modulates the gradient of $\|\Psi(p) - o_{I^*(p)}^*\|_1$ with a weight $w(p)$ taking into account the ratio of embedding quality – how close a predicted embedding is to the annotated center of the mass of the corresponding instance – to the instance width – the smallest distance between the center of mass of the corresponding instance and that of other instances.

Let $I^*(p) \in \{1, \dots, |\mathcal{I}^*|\}$ be the annotated instance label of pixel p . Hence, $o_{I^*(p)}^*$ is the center of the annotated instance to which pixel p belongs. We define $\hat{o}_i^* = \operatorname{argmin}_{j \neq i} \|o_i^* - o_j^*\|_2$ as the center of the closest instance to instance i , *i.e.*, the centroid constraining the bandwidth of instance i the most. The weight $w(p)$ is then defined as:

$$w(p) = \mathbb{1}_{\{p \in \mathcal{F}^*\}} \left(1 - \exp\left\{-\frac{1}{2\sigma_{I^*(p)}^2} \|\Psi(p) - o_{I^*(p)}^*\|_2^2\right\} \right), \quad (6.4)$$

where

$$\sigma_i = \sqrt{\frac{\|o_i^* - \tilde{o}_i^*\|_2^2}{8 \ln 2}}, \quad \forall i, \quad (6.5)$$

is the instance width parameter defined such that, in Equation (6.4), $w(p) = 0.5$ when $\|\Psi(p) - o_{I^*(p)}^*\|_2 = \|o_{I^*(p)}^* - \tilde{o}_{I^*(p)}^*\|_2/2$.

While $w(p)$ depends on the pixel’s offset vector $\Phi(p)$ through $\Psi(p) = \Phi(p) + p$, $w^\times(p)$ is seen as a constant by the optimization process, *i.e.*, when computing the gradients of Equation (6.3) with respect to the predicted offset. An example of ADES is illustrated in Figure 6.5. For two embeddings belonging to the same instance, *i.e.*, fixed σ_i in (6.4), ADES adjusts the optimization step in favor of the erroneous embedding, *i.e.*, with larger $\|\Psi(p) - o_{I^*(p)}^*\|_2$. Moreover, for two embeddings with the same absolute error $\|\Psi(p) - o_{I^*(p)}^*\|_2$, the one belonging to an instance with smaller bandwidth σ_i gets larger optimization step.

The proposed loss function shares similar motivations as some of previous works [LGG⁺17, NBPG19]. Focal loss [LGG⁺17] applies to classification task and multiplies the cross-entropy loss by a factor to focus on learning misclassified examples. Unlike ADES, the Focal loss is not intended to re-weight the gradient. Neven *et al.* [NBPG19] propose a loss supervising instances based on their width in the embedding space. However, instance width is predicted by the network and optimized jointly with the offset vectors, which result in a more complex network architecture and supervision mechanism compared to that of Panoptic-Deeplab.

6.3 Synthetic experiments

In this section, we will investigate the capability of Panoptic-Deeplab to reconnect instance parts depending on the data it has been trained on, as well as the capability of metrics to measure improvements on dealing with partitioned instances. Those experiments will help us to study to what extent Panoptic-Deeplab is able to exploit color information to reconnect instances, *i.e.*, deal with hard occlusions. We also study the extent to which the method is able to generalize to object shapes that were not encountered during training.

Setup. For each ISMNST dataset, 500 images with 512×256 pixels are generated. The center of clusters are spaced by 64 pixels in both directions, resulting in a 7×3 grid of clusters covering the image. For each cluster,

Train \ Eval.	<i>spaced</i>	<i>packed</i>	<i>partitioned</i>
<i>spaced</i>	97 –	40 –	10 3
<i>packed</i>	98 –	97 –	72 35
<i>partitioned</i>	92 –	94 –	87 79

Table 6.1: IS quality (PQ|PIQ, in %) achieved from training a model on a dataset and testing it on another dataset. PIQ is undefined for datasets without partitioned instances.

instances are pasted by (uniformly at random) positioning their center of masses in a 64×64 window centered at the center of that cluster. Those settings lead the images to appear uniformly populated except at borders. The maximum number of instances and tries per cluster is set to 20 and 1000, respectively.

All of the following experiments are based on the public re-implementation of Panoptic-Deeplab by its authors[CCZ⁺20] augmented with Modular Output Mathematics [Ist21] from Chapter 3. Each experiment consists of a 300 epochs training on an Nvidia V100 32GB GPU with batch size of 8. We use the Panoptic-Deeplab configuration based on the HRNet-W48 backbone [WSC⁺19] but do not use pretrained weights. We set the learning rate to 10^{-2} to adapt to the lower batch size and single GPU. For grayscale images, the learning rate is set to 10^{-3} . The original L1 loss is kept. On every iteration, input images are upsampled to 1536×768 pixels with a slight scale data augmentation. The mean and variance used for input normalization are computed on the *packed* versions of colored and grayscale ISMNIST.

6.3.1 Benefit of hard occlusions during training

In this first set of experiments, we investigate whether Panoptic-Deeplab has any ability to reconnect partitioned instances, and if so, whether it emerges naturally from training on isolated instances. To this end, we train independent models on three training sets from ISMNIST explained in Section 6.2.2, *i.e.*, *spaced*, *packed* and *partitioned*, and evaluate their performance on every test sets. To avoid any ambiguity when reconnecting partitioned instances, *i.e.*, to render the task as simple as possible for the network, the digits are rendered with different (random) colors.

Results. Cross-dataset PQ and PIQ metrics are provided in Table 6.1. Models trained on *spaced* and *packed* datasets clearly fail to reconnect in-

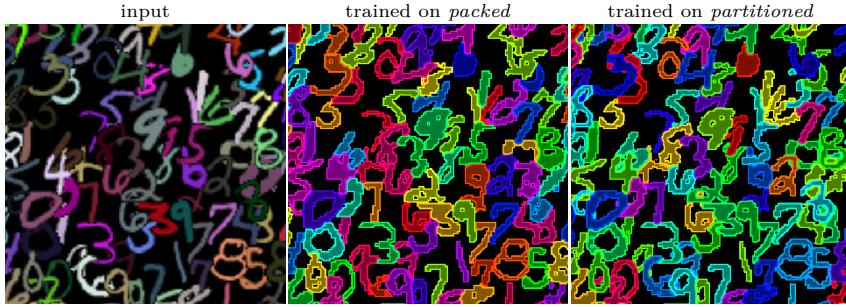


Figure 6.6: An example extracted from the last column of Table 6.1. An input image (left) taken from the test split of *partitioned* dataset is fed to the models trained on the *packed* (middle) and *partitioned* datasets. The results of the model trained on *spaced* dataset is not shown here, as it failed to segment any of the instances.

stances in *partitioned*. However, when trained on *partitioned* dataset, the network achieves much higher performance. Figure 6.6 provides an example image (left) processed by the models trained on *packed* (middle) and *partitioned* (right) datasets. The latter suggests that performance close to visual perfection is possible.

Moreover, those results show that Panoptic-deeplab is clearly capable of handling partitioned instances when they are frequent in the training set. However, even when instance partitions have similar color content, a network that has not seen partitioned instances during training will not be able to reconnect them.

6.3.2 Benefit of more partitioned instances during training and evaluation

In this experiment, we use the *partitioned_f* datasets from ISMNIST and explore the impact of the frequency of partitioned instances during training on the performance of the model during testing phase.

We use the strategy explained in Section 6.2.2 to generate *partitioned_f* datasets with $f \in \{0, \frac{1}{100}, \frac{1}{30}, \frac{1}{10}, \frac{1}{3}, 1\}$ and use their training sets during training phase. Evaluation is performed on their test sets with $f \in \{\frac{1}{10}, 1\}$.

Results. Evaluation curves on *partitioned* dataset are provided in black in Figure 6.7. Both PQ and PIQ increase monotonically with f . Thanks to

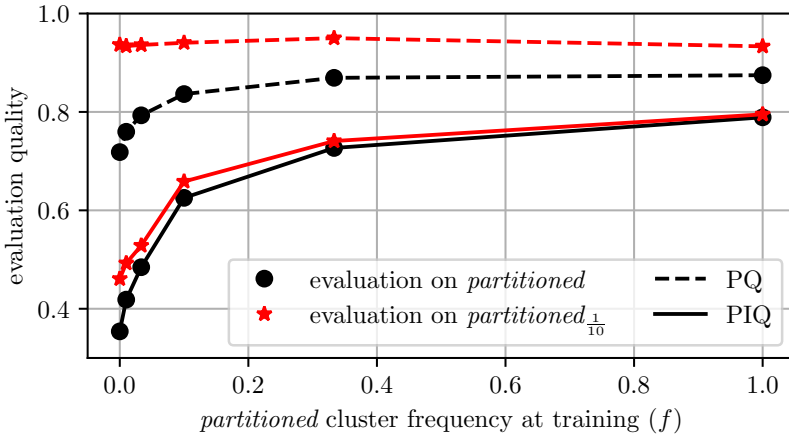


Figure 6.7: Performance of IS task improves on *partitioned* and *partitioned*_{1/10} as the frequency of partitioned instances seen during training increases. PQ is agnostic to the partitioned instances, unless they are dominant in the evaluation set.

PIQ metric, it can be seen that about one third of the secondary partitions are properly assigned to their instance centroid for the model trained on *packed* dataset. Training on *partitioned*_{1/10} dataset improves that percentage by a factor of two and reduces more than half of the gap between PQ and PIQ metrics. The same metrics are reported when evaluating the same models on *partitioned*_{1/10}; see red curves in Figure 6.7.

Additionally, the results illustrate that PQ fails to capture progress on hard examples when the majority of cases are isolated instances or easy occlusions; similar to most available datasets. The PIQ metric, in contrast, reports a very similar trend to the one obtained in Figure 6.7 when hard cases were more frequent in the test set.

6.3.3 Benefit of color content for IS task

In this section, we study whether Convolutional Neural Networks (CNNs) are able to reconnect partitions of objects on the *partitioned* dataset without the use of discriminative color content. All instances are now imprinted using white color.



Figure 6.8: An example grayscale input image (left) from test split, the output of a model trained on grayscale images (middle), and the ground-truth labels.

Colored instances			Grayscale instances		
Train \ Eval.	Even	Odd	Train \ Eval.	Even	Odd
Even	88 79	88 81	Even	86 71	81 63
Odd	86 72	89 84	Odd	72 54	88 79

Table 6.2: Cross-training. Metrics (PQ|PIQ, in %) achieved from training a model on a given dataset and testing it on another dataset.

Results. The model is able to solve the problem in an impressive manner. It achieves a PQ of 86% and a PIQ of 73%, which are, respectively, 1% and 6% below what was achieved with models trained on colored inputs. Therefore, the network is able to reconnect partitions, despite the fact that the reconnection is much less obvious than in the colored setup. See Figure 6.8 for an example.

6.3.4 Benefit of shape diversity during training

We investigate to what extent object shapes seen during training affect the ability of the network to reconnect partitions. In the previous part, we observed that CNNs are able to reconnect remote parts without color cues. We now hypothesize that they must rely on a learned set of identifiable shapes rather than some more profound internal reasoning.

To this end, we train the network on an ISMNIST dataset composed of *even* (and *odd*) digits and evaluate it on another dataset with *odd* (respectively, *even*) digits. This experiment is done in both colored and grayscale modes.

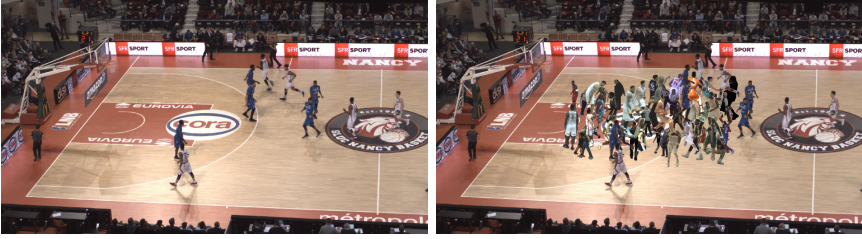


Figure 6.9: Example image augmentation obtained with our data augmentation strategy, from the image on the left. We copy-pasted 45 annotated instances, without augmentation. At least 50% of the pixels from each pasted instance stay visible.

Results. Table 6.2 reports the performance obtained for colored and grayscale datasets. We make three key observations. First, the PQ of all models when trained and tested on the same dataset (on the diagonals) is quite similar. However, the PIQ metric suggests that the *even* dataset is harder for IS, because even digits have more pixels at the extremes and create more difficult occlusion patterns. Therefore, training a network on *even* digits and testing it on *odd* digits performs better than the other way around. Second, by comparing the antidiagonals of the two tables, it appears that the network makes relatively good use of the color content to generalize to unseen shapes when it is available. Third, the performance systematically drops when considering a model trained on the other digits, even when color content is available. This suggests that the network relies on objects structures seen during training; hence, it is confused when encountering unseen structures.

6.4 Experiments on a real dataset

In this section, we demonstrate that the proposed framework helps in a real-life use-case involving humans, meaning non-trivial background and complex dynamic patterns. The DeepSport dataset [GZI⁺21], introduced in Section 2.1.2, is a relevant choice for this analysis since it presents significant instances deformations, a large variety of patterns, but also strong similarities between instances in a same image. In this dataset, occlusions are frequent but far from being systematic.

Setup. Original images have a resolution of 1024×768 pixels. To increase the level of occlusions in the training set, we follow the strategy described in Section 6.2.2 to add three clusters of size 128×128 pixels, each with 15 instances, aligned horizontally at mid-height in the image. We copy-paste (with no augmentation) instances that are randomly selected among the delineated ones across the whole training set, while overlaying the original instances. Pasted instance centroids are prevented from overlapping in the centroid heatmap by enforcing a minimal distance of 16 pixels with any other centroid. Their masks are guaranteed to comprise at least 50% of the pixels of the original mask. In contrast to ISMNIST, however, copied instances may already be occluded. An example is provided in Figure 6.9. The training setup is the same as in Section 6.3, except here we used the ImageNet-pretrained backbone[WSC⁺19] and trained the full network for 450 epochs with a learning rate of 10^{-2} on images of size 1024×768 with a slight scale and color jitter data augmentation at each iteration.

Results. The 6 first rows of Table 6.3 provide quality of IS on both augmented and unaugmented datasets with L1, SL1, and ADES loss functions. Firstly, we notice that less than half of disconnected pixels are correctly re-assigned to their instances without data augmentation, even with oracle segmentation, demonstrating the difficulties of the embeddings in those cases. Secondly, data augmentation improves PQ, PIQ, and PIQo metrics. This confirms the claims of [DMS18, GCS⁺21] that the unrealistic artifacts introduced by copy-pasting do not overweight the benefits when performance on real data is considered. Thirdly, ADES loss function yields highest PIQ and PIQo values. Fourthly, combination of ADES – or, to a lesser extent, SL1 – with data augmentation achieves the best reconnection ability of partitioned instances. This reveals the complementarity between our proposals. In particular, the benefit of modulating the gradient of loss function is conditioned to the presence of sufficient relevant samples. Finally, the PIQo values suggest that when imperfect semantic segmentation is ignored, the ability of the base-line Panoptic-Deeplab for segmenting partitioned instances increases by 10% in PIQo by only leveraging data augmentation during training. Moreover, we gain an additional 8% improvement in PIQo by coupling the proposed ADES loss function, contrasting heavily with its 2% contribution when the latter is considered alone.

Connection with Chapter 5. While we made the choice to focus on fixed spatial embeddings with Panoptic-Deeplab in this chapter, we intro-

Augmented training set	Loss	PQ	PIQ	PIQ _o
no	L1	83.16	38.10	45.37
	SL1	82.97	36.95	44.27
	ADES	83.50	39.36	47.56
yes	L1	85.30	44.47	55.32
	SL1	84.02	49.80	57.37
	ADES	84.21	52.13	63.62
no	C3-CCP	83.99	43.08	52.64
yes	C3-CCP	85.37	60.29	70.49

Table 6.3: Performance of Panoptic-Deeplab on DeepSport dataset when trained with and without data augmentation and with different loss functions. The two last rows exhibit results obtained with hybrid embeddings from Chapter 5 (2-D spatial embedding concatenated with 1-D associative embedding).

duce hybrid embeddings in Chapter 5 as a solution to enable the handling of instances with completely overlapping centroids. We argue there that they result in more capabilities for decoding them, but also in better supervision as the network is given more incentive to differentiate them. Table 6.3 confirms these intuitions as hybrid embeddings (C3-CCP rows) prove to be the fittest solution we proposed to handle occlusions with, and without data augmentation. Interestingly, with data-augmentation, the L1 and C3-CCP methods achieve very close PQ, while the latter reconnects 70.5% of disconnected pixels to their main components, against 55.3% for L1.

6.5 Conclusion

We showed that the frequency and variety of occluded instances during training were crucial for Panoptic-Deeplab to handle them. To this end, we introduced ISMNIST, a family of synthetic IS datasets based on MNIST with controllable color, shape, and occlusion population settings. That led us to use similar data augmentation on a real dataset by increasing the number of occluded instances. We showed that improvements in segmenting partitioned instances are hardly measurable using classical metrics; resulting in introduction of a novel PIQ metric tailored to this end. Moreover, we proposed a loss function, ADES, adjusting the optimization step in favor of ambiguous predicted embeddings, which showed significant improvement in segmentation quality of partitioned instances when they are sufficiently

present in the training dataset.

We believe this work sheds lights on solving the hardest IS cases on datasets with relatively few occlusions in the training set, illustrating the kind of improvement that can be reached by playing on different components. Coupled with the best method obtained in Chapter 5, we go from reconnecting 45% of remote pixels on the DeepSport dataset to 70%.

Possible future investigations comprise *(i)* comparing the data-efficiency with respect to occlusions, *(ii)* disentangling the contributions of quantity and frequency of partitioned instances and *(iii)* applying the same tools on multi-class datasets and, in particular, assessing whether context modeling is necessary or not to observe the same gains [DMS18, FSW⁺19, GCS⁺21].

Chapter 7

Conclusion

Now has come the time to conclude this thesis. Overall, we can say that our objective to obtain a robust end-to-end IS method capable of resolving occlusions has been met.

Indeed, we have shown that the strengths of spatial and associative embeddings could be combined in a hybrid IS method that is very generic, *i.e.*, not relying on any prior such as human pose or projective geometry, and theoretically sound to deal with the strongest occlusions. However, the data on which this method has to be trained has to exhibit a significant portion of occlusions in order for the method to resolve them. Hopefully, data-augmentation seems able to address the problem of scarce occlusions in training data.

We recapitulate the main lessons of each chapter in Section 7.1, and then discuss several related research tracks that we believe to be promising in Section 7.2.

7.1 Main lessons

In Chapter 2, we introduced the IS task and the types of models that are used to solve it, with a particular attention devoted to the Non-Maximum Suppression (NMS) sampling step systematically involved in IS decoding. We have discussed that bottom-up methods, being free from the sampling of bounding boxes, seem better suited to deal with occlusions than top-down methods [HGDG17]. However, each bottom-up method has its own limitations. For instance *(i)* associative embeddings [NHD17] require the

emergence of structure in the embedding space at training, require tedious decoding and may require the network to avoid translation-equivariance, *i.e.*, to avoid working locally, while *(ii)* spatial embeddings with fixed centroids [CCZ⁺20] are limited in their handling of occlusions. *(iii)* spatial embeddings with learnable centroids [NBPG19] alleviate the problem of close centroids by allowing to move them in the embedding space, but again at the expense of training and decoding difficulties.

In Chapter 3, we developed software engineering constructs that helped us to experiment in a flexible manner with the different formulations of embedding-based methods. We showed that classes and multiple inheritance features available in object-oriented languages could help us to achieve the software equivalent of combining incomplete equation systems as those we formalize in Chapters 5 and 6, allowing for a modular implementation of Output Mathematics (OM), which we call ModulOM. Integrating ModulOM to the code base of Panoptic-Deeplab [CCZ⁺20], a very well established baseline for instance and panoptic segmentation, helped us to confirm the benefits and negligible overhead of our proposal, while also providing us a strong setup for future experiments.

In Chapter 4, we tackled the team discrimination problem using associative embeddings, turning each team into an instance and applying a framework similar to that of [NHD17]. This experiment helped us to confirm that the network could agree on embeddings on disconnected regions across the whole image and that we were able to decode them efficiently. Despite the low quality of the annotations, the model also generalized seamlessly to unseen teams and arenas.

In Chapter 5, we present a unified mathematical framework integrating a significant set of interdependent components coming from the literature, in particular, embedding and centroid definitions, supervisions and decoding strategies, mainly coming from the methods described in Chapters 2 and 4. Implementation-wise, this resulted in the expansion of the sole ModulOM component integrated in Chapter 3. Experiments on associative embeddings allowed us to better understand and solve the different challenges behind the convergence of the embedding space, and to design and compare fast decoding strategies. The best results are obtained with Centroid-Centroid Push (CCP) [DBNVG17] and the Learned Quality Seed Map (LQSM) [NBPG19], where the network learns to estimate the quality of its predictions. Although promising, the obtained performance was still slightly inferior to that of fixed spatial embeddings [CCZ⁺20] (by 2% PQ), which seemed reasonable to attribute to incompatibility of as-

sociative embeddings with translation-equivariance. We capitalized on the knowledge gathered on associative embeddings to propose hybrid embedding methods that would leverage translation-equivariance from spatial embeddings and local discriminability of associative embeddings. Equipped with CCP+LQSM, those proved to outperform spatial embeddings, lifting the latters' theoretical limitations and allowing for more guarantees on the supervision and decoding of overlapping instances. The improvement in performance measured by metrics on real data, however remained limited.

In Chapter 6, we showed through experiments on synthetic datasets and dedicated metrics that the ability of IS methods to deal with partitioned instances, *i.e.*, instances split because of occlusions, does not emerge from simpler examples and instead depends heavily on the frequency and variety of partitioned instances at training. We also illustrated that common metrics, *e.g.*, Panoptic Quality, could fail to report such improvements. We then demonstrated on real data that data-augmentation was an effective solution to increase the performance of partitioned instances artificially, also multiplying the impact of other measures to tackle them, *e.g.*, online hard example mining. In this context, hybrid embeddings proposed in Chapter 5 prove to become particularly strong to handle occlusions and reconnections.

7.2 Perspectives

As many research works, this thesis brought up more questions than it answered. We survey here the main questions that subsist and may inspire future research, along with some more speculative tracks.

Modular Output Mathematics (ModulOM). ModulOM introduced in Chapter 3 consists in a way of implementing the OM in a modular fashion. However, its reliance on multiple inheritance may introduce a barrier for the computing library, *e.g.*, PyTorch, Tensorflow, to optimize the code for optimal throughput. We believe that this would be an important issue to be resolved before pushing for more widespread adoption. It is also uncertain what mechanisms should be leveraged in languages that do not implement multiple inheritance, *e.g.*, C++ and Java.

Hybrid embeddings. We believe that the hybrid embedding framework developed in Chapter 5 could be used to tackle different tasks, *e.g.*, action spotting, object detection, object tracking, while being particularly resistant

to occlusions. However, adapting it to each of those tasks may bring its own challenges. In particular, some tasks may require the introduction of new components, whose interactions with the proposed framework have not been studied yet.

Additionally, hybrid embeddings could help solve many tasks in parallel, *e.g.*, simultaneous semantic segmentation, IS, dense pose estimation and action recognition. Joint formulation and supervision can be achieved in very different ways and would probably benefit from additional research.

Loss design. Chapters 5 and 6 demonstrate the importance of loss design, *e.g.*, convergence of targets that depend on network outputs, hard example mining. We believe that guidelines could be drawn from the systematic comparison of intermediate losses between CCP over DPPES created with the help of gradient programming, presented in Appendix A.2.

Data biases. Chapter 6 highlights the benefits of biasing training data towards harder examples. We have seen that a naive instance copy-pasting could provide a suprising boost with respect to the handling of occlusions on the DeepSport dataset. Several questions however remain: How should we dose the augmentation so as to maximize performance on the original data? Is a naive data augmentation sufficient on datasets containing objects with different classes, scales or topologies? What is the impact of violating priors that hold on natural data, *e.g.*, perspective and semantic context?

Robustness analysis. We discuss in Section 2.3 reasons why we believe top-down methods to be less appropriate to handle occlusions than bottom-up methods. Studying their robustness on different datasets would be valuable to the literature. This could be done, for instance, by quantifying the uncertainty of the bounding box prediction and the sensitivity of the segmentation with respect to the quality of the bounding box estimation. Analogously, the impact of adding noise to the network outputs on the robustness of bottom-up methods could be measured. Evaluating the robustness against noise applied on the input signal could also be of interest.

Architectures for IS. Our research was purposefully focusing on CNNs as they matched the pixelwise paradigm of IS with simplicity but comprised interesting challenges to overcome, *e.g.*, the alignment between decoding and supervision investigated in Chapter 5. Transformer layers [VSP⁺17]

are becoming increasingly used in vision [DBK⁺21, CMS⁺22] as they have the potential to replace convolutional layers in many places [DLLT21], and perhaps even to relief methods from the need for NMS [LXZ⁺23]. While transformers are appealing, their impact on IS quality might not necessarily be obvious when taking into account additional computational cost and required training data.

Continuous improvement, zero-shot learning, and sketching. Increasing attention is devoted to continuously improving systems, *i.e.*, systems that can adapt using data acquired and annotated as they are used, *e.g.*, from human corrections on erroneous predictions. On one end of the spectrum, active learning [LWR⁺22] consists in continuously training a complete model on a growing training set. On the other end of the spectrum lies zero-shot learning, *i.e.*, the resolution of new tasks defined from a small set of examples as context at inference [LEB08].

There is growing evidence that the transformer architecture enables zero-shot learning, *e.g.*, in segmentation [KMR⁺23]. This would allow human experts in a task to better guide a pretrained generic model to fit their needs, and to keep the sensitive examples private. However, as a large number of examples increases model runtime and memory usage, the selection of a small but diverse set of examples to guide the model becomes a concern.

Halfway between active and zero-shot learning, concepts similar to dataset sketching [GCK⁺21], *i.e.*, providing a compressed representation of a potentially growing set of examples, could help guiding generic models in more principled ways, and lowering the hardware requirements for continuously improving models. Although theoretically appealing, many questions arise with respect to the training of such general models, *e.g.*, the data distribution biases to adopt, to the sketching strategy to use, and how this would compare to active learning and low-rank fine-tuning strategies [HysW⁺22].

List of publications

1. **Maxime Istasse** and Christophe De Vleeschouwer. Spatio-Associative Embeddings for Instance Segmentation in Complex Scenes. 2023. *In preparation for computer vision journal submission.*
2. **Maxime Istasse**, Amirafshar Moshtaghpour, and Christophe De Vleeschouwer. Resolving Occlusions when Segmenting Instances with Spatial Embeddings. 2023. *In preparation for computer vision conference submission.*
3. Gabriel Van Zandycke, Vladimir Somers, **Maxime Istasse**, Carlo Del Don, and Davide Zambrano. Deepsportradar-v1: Computer vision dataset for sports understanding with high quality annotations. In *the ACM International Conference on Multimedia, MMSports Workshop*, pages 1–8, 2022.
4. **Maxime Istasse**, Kim Mens, and Christophe De Vleeschouwer. ModulOM: Disseminating deep learning research with modular output mathematics. In *the International Conference on Learning Representations (ICLR), Beyond Static Papers Workshop*, 2021.

5. Seyed Abolfazl Ghasemzadeh, Gabriel Van Zandycke, **Maxime Istasse**, Niels Sayez, Amirafshar Moshtaghpour, and Christophe De Vleeschouwer. Deepsportlab: a unified framework for ball detection, player instance segmentation and pose estimation in team sports scenes. In *the British Machine Vision Conference (BMVC)*, page 379, 2021.
6. **Maxime Istasse**, Julien Moreau, and Christophe De Vleeschouwer. Associative embedding for team discrimination. In *the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2477–2486, 2019.
7. Adrien Delière, Ashwani Kumar, **Maxime Istasse**, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. Ordinal pooling. In *the British Machine Vision Conference (BMVC)*, pages 1–12, Cardiff, Wales, September 2019.
8. Anthony Cioppa, Adrien Delière, **Maxime Istasse**, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2505–2514, Long Beach, California, USA, June 2019.

Appendix

A.1 NMS and its implementations

There is a small discrepancy between the results of the formal definition of NMS we introduced in Equation (2.11) and the actual sequential algorithms described by [GDDM14, BSCD17, NBPG19] and Section 2.4.3 to process seed maps for predicted embeddings $\Psi(p) \in \mathbb{R}^D$ and bounding boxes $B_i \in \mathbb{R}^4$.

In the formal NMS, an element $e \in \mathcal{E}$ can be cancelled by another cancelled element e' as long as e' has a higher score and is in e 's neighborhood, this is what allows the NMS_P of a heatmap with a small window size, *e.g.*, 3×3 , to only extract local maximas [CCZ⁺20]. Additionally, such a spatial NMS is convenient to implement spatially as it only requires one max-pooling operation and one comparison.

However, NMS_Ψ on predicted embeddings and NMS_B on predicted bounding boxes are almost always implemented using a sequential algorithm that does not allow cancelled elements to cancel other elements by themselves, hence corresponding to a relaxed NMS. [BZXL19] is the only work we are aware of that truly performs NMS_b with Equation (2.11), and reports a minor drop in performance. This suggests that in those spaces, the elements may be pruned too aggressively if following the definition.

To summarize, most implementations of NMS_Ψ and NMS_B deviate slightly from the NMS definition from Equation (2.11), leading them to a minor **increase** in performance. In our implementation of NMS_Ψ , we therefore keep using the relaxed NMS, *i.e.*, the sequential algorithm, *e.g.*, described in Section 2.4.3, while leaving notations in this dissertation unchanged.

A.2 Gradient programming

To allow more modular phrasing and more understandable interactions during supervision, we create pseudo-losses in Sections 5.3.2 and 6.2.3 where the direction of the gradient with respect to network outputs and its magnitude are separately controllable. To this end, we build pseudo-losses from functions $D(p)$ specifying the gradient direction function, with unit-magnitude gradient with respect to network outputs, *e.g.*, the L2 loss $\|\Psi(p) - o_{I^*}^*(p)\|_2$, and from a weight $w(p)$ whose gradient is truncated. To denote gradient truncation, we introduce the gradient truncation symbol “ $\times$ ”. In our case, the final loss to compute for gradient backpropagation can be expressed as $L = \frac{1}{w^\times(p)} \sum_p w^\times(p) D(p)$. While there is a mismatch between $w(p)D(p)$ and the function being optimized when $w(p)$ depends on $\Phi(p)$, this is intended to provide better control over optimization steps.

Truncating the gradients and relying on such pseudo-losses may seem inadequate. However the fundamental qualities of loss functions being optimized, *e.g.*, convexity, continuity, derivability and gradient, get diluted as mini-batch SGD is adopted and optimizes averaged or reweighted quantities, in our case from multiple sublosses. Hence, it may be that the gradient direction and magnitude is everything at play to guide the optimization process. Additionally, we remind that most networks never reach global minimas, but instead suboptimal solutions whose gradients cancel each other. Gradient programming may convey a more direct control over this equilibrium.

As a first example, which we frame outside of our embedding-based framework, we can mimic the behavior of the Smooth L1 (SL1) loss, whose gradient fades linearly when a prediction $y \in \mathbb{R}$ reaches the ground-truth $t \in \mathbb{R}$. To achieve this, SL1 is quadratic in $[t - \beta, t + \beta]$ and otherwise linear:

$$\text{SL1}(y, t) = \begin{cases} 0.5|y - t|^2/\beta & \text{if } |y - t| < \beta \\ |y - t| - 0.5\beta & \text{otherwise} \end{cases} \quad (\text{A.1})$$

The same supervision can be obtained through pseudo-SL1 (pSL1) obtained with gradient programming:

$$\begin{cases} \text{pSL1}(y, t) = w^\times(y, t) D(y, t) \\ D(y, t) = |y - t| \\ w(y, t) = \min(|y - t|/\beta, 1) \end{cases} \quad (\text{A.2})$$

Thanks to $D(y, t)$, it is crystal clear that the goal is to obtain $y = t$. The

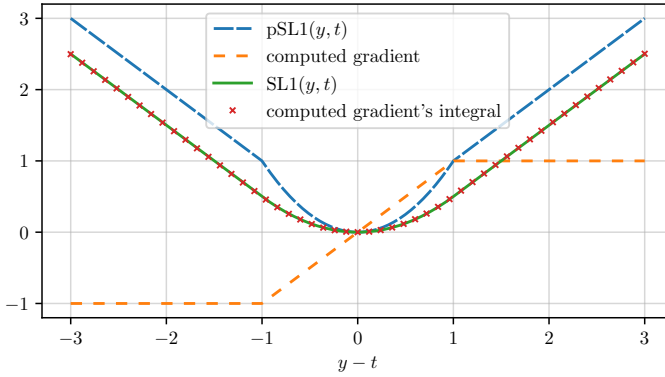


Figure A.1: Illustration of the mismatch between reported and optimized functions when using gradient programming: the gradient computed from pSL1 is not that of pSL1, but rather that of SL1. Hence, in a training setup, the function being optimized would be SL1. β in this case is set to 1.

gradient weight $w(y, t)$ makes it easier to play on the decay of the gradient, which could for instance achieve a Gaussian-shaped gradient magnitude, that would otherwise be much harder to obtain. When y and t are multi-dimensional, $D(y, t)$ can also be practical to choose how the gradient is distributed among the axes, *e.g.*, switching $D(y, t)$ to $\|y - t\|_2$ or $\|y - t\|_1$ will achieve a different balance. In Section 5.3.2, we also play with the direction to overcome challenging convergence problems.

Regarding limitations of gradient programming, it is important to mention that the computed function, here pSL1(y, t), does not provide the value of the function being optimized, *i.e.*, SL1(y, t). The observed values during training may in fact be meaningless. This mismatch is illustrated in Figure A.1. We therefore use gradient programming when we care more about controlling the optimization steps than the actual value of the loss being optimized.

A.3 Associative embedding convergence under centroid-pixel push-pull supervision

So as to illustrate the differences between the different centroid-pixel supervisions proposed in Section 5.3, we performed experiments in pure optimization setups: embeddings are autonomous parameters directly optimized to minimize L_{emb} via gradient descent. Figures A.2 to A.4 illustrate the convergence dynamics of PP1, PP2 and DPPEs with varying number of instances, number of points, and embedding dimensionality.

It can be observed in Figure A.2 that all losses do not converge at the same rate, and that increasing number of instances and points delay convergence. Figure A.3 illustrates that all losses finally converge in the 2-D case, although some methods result in smoother centroid trajectories during training. Overall, DPPEs reaches solutions quicker and with smoother centroids trajectories. In 1-D, as illustrated in Figure A.4, only DPPEs is able to shape an embedding space that is convincing in all scenarios.

While the system dynamics differ in a setup involving the training of neural network, chances are that the more consistent feedback provided by DPPEs eases the convergence of the neural network and its outputs.

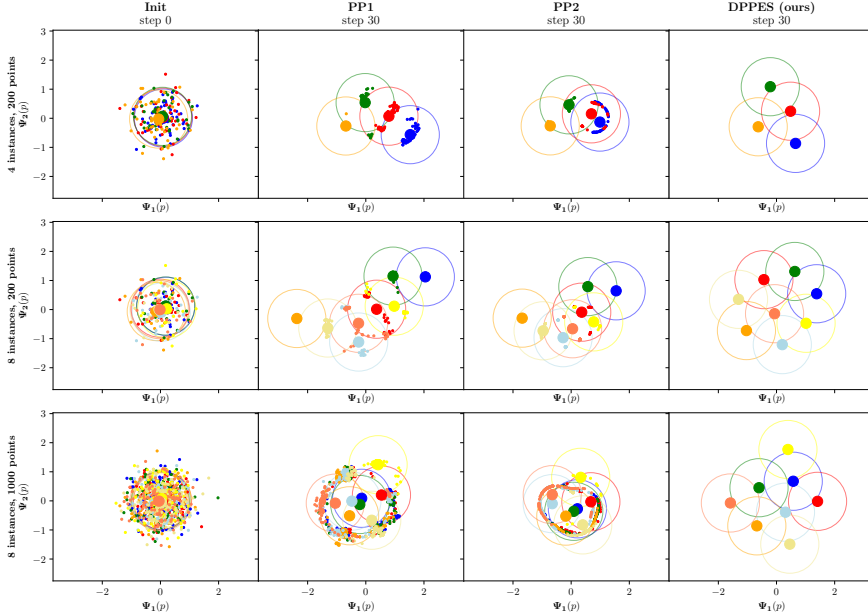


Figure A.2: Snapshots of randomly initialized 2-D embeddings being adjusted to minimize PP1, PP2 and DPPES losses from Section 5.3, decoupled from any neural network training. 30 gradient descent steps have been performed from 3 initial configuration (on the left). Small dots represent embeddings, each assigned to an instance represented by a color. Each bigger dot represents an instance centroid, *i.e.*, the center of mass of small dots of the same color. The first row is a configuration with 200 points split into 4 instances. The second row configuration contains keeps 200 points, but split in 8 instances. The third row keeps 8 instances, but with 1000 points. In each case, DPPES has fully converged within those 30 optimization steps, while PP1 and PP2 are still in the process. It can also be seen that increasing the number of instances or the number of points renders the task more complex for them.

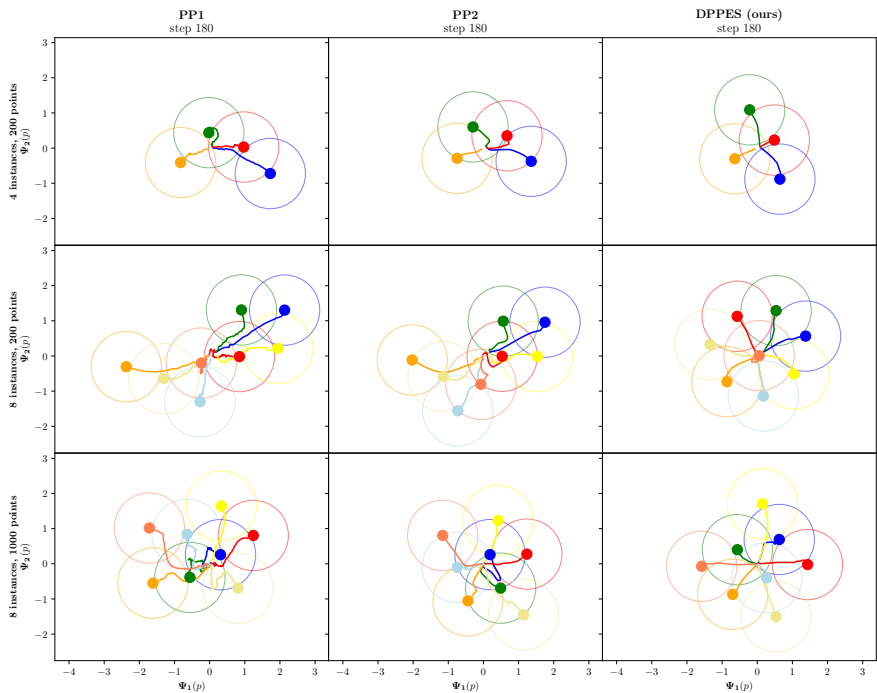


Figure A.3: Illustration of the final instance centroids and their trajectories after adjusting 2-D instance embeddings to minimize PP1, PP2 and DPPES losses, decoupled from any neural network training. 180 gradient descent steps have been performed from 3 initial configurations (the same as in Figure A.2). It can be observed that all PP1, PP2 and DPPES reach optimal solutions, while DPPES achieves smoother trajectories.

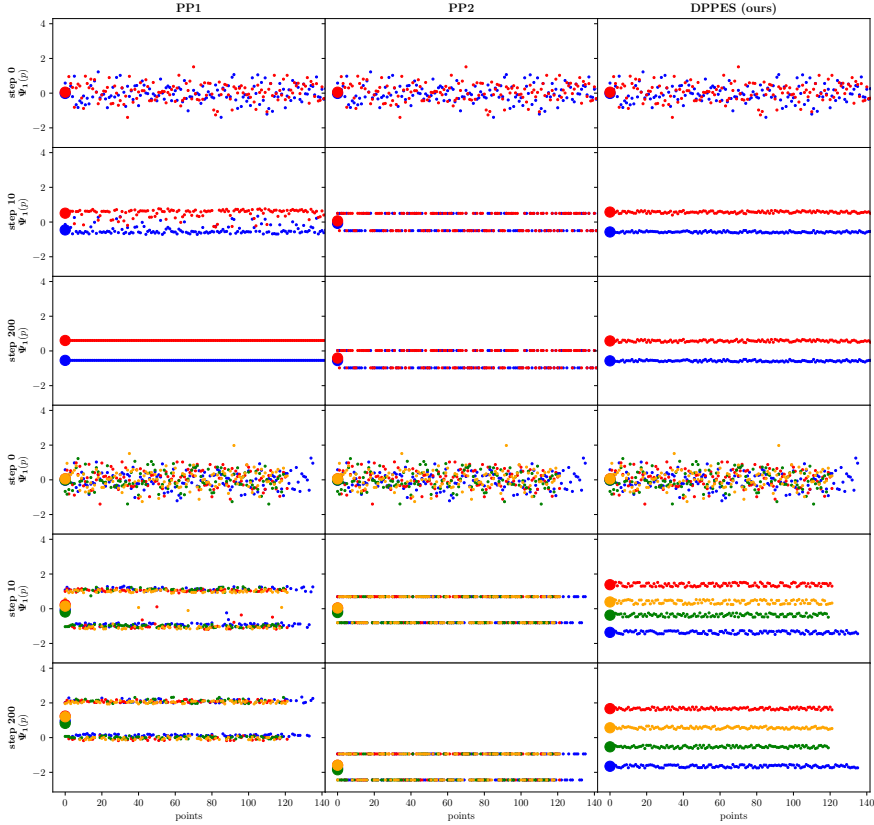


Figure A.4: Snapshots of randomly initialized 1-D embeddings being adjusted to minimize PP1, PP2 and DPPES losses, decoupled from any neural network training. Initial embeddings and those achieved after 10 and 200 optimization steps are provided for 2 configurations: the first with 500 points split in 2 instances (3 first rows), and the second with 500 points split in 4 instances (3 last rows). We can observe that both PP1 and DPPES converge with 2 instances, while only DPPES manages to converge with 4 instances. In the other cases, we can attest that convergence fails as the states at 200 steps are almost identical to those reached after 10 steps.

Bibliography

- [AJBV11] Alexandre Alahi, Laurent Jacques, Yannick Boursier, and Pierre Vanderghenst. Sparsity driven people localization with a heterogeneous network of cameras. *Journal of Mathematical Imaging and Vision*, 41(1):39–58, Sep 2011.
- [BC90] Gilad Bracha and William Cook. Mixin-based inheritance. In *Proceedings of the European Conference on Object-Oriented Programming on Object-Oriented Programming Systems, Languages, and Applications*, OOPSLA/ECOOP '90, page 303–311, New York, NY, USA, 1990.
- [BCH⁺96] Kim Barrett, Bob Cassels, Paul Haahr, David A. Moon, Keith Playford, and P. Tucker Withington. A monotonic superclass linearization for Dylan. In *the ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA)*, pages 69–82, San Jose, California, United States, 1996.
- [BJ06] David M. Blei and Michael I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 03 2006.
- [BKC17] 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, Dec 2017.
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

- [BLC⁺14] Alina Bialkowski, Patrick Lucey, Peter Carr, Sridha Sridharan, and Iain Matthews. *Representing Team Behaviours from Noisy Data Using Player Role*, pages 247–269. Cham, 2014.
- [BPL10] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *the International Conference on Machine Learning (ICML)*, pages 111–118, 2010.
- [Bru02] Kim B. Bruce. *Foundations of Object-oriented Languages: Types and Semantics*. 2002.
- [BSCD17] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS - Improving Object Detection with One Line of Code. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5562–5570, October 2017.
- [BTB18] Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. The iou-loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. pages 4413–4421, Salt Lake City, UT, 2018.
- [BZXL19] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT: real-time instance segmentation. In *the IEEE International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019.
- [CCZ⁺20] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12472–12482, 2020.
- [CDI⁺19] Anthony Cioppa, Adrien Delière, **Maxime Istasse**, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2505–2514, Long Beach, California, USA, June 2019.

- [CDVD18] Anthony Cioppa, Adrien Deliege, and Marc Van Droogenbroeck. A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games. In *the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, June 2018.
- [Cho15] François Chollet. Keras. <https://keras.io>, 2015.
- [CLC⁺16] Jianhui Chen, Hoang M. Le, Peter Carr, Yisong Yue, and James J. Little. Learning online smooth predictors for real-time camera planning using recurrent decision trees. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [CMS⁺22] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. 2022.
- [COR⁺16] 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 *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.
- [CPK⁺17] 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.
- [CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. [arXiv:1706.05587](https://arxiv.org/abs/1706.05587).
- [Csá01] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*, 24(48):7, 2001.
- [CSM12] Peter Carr, Yaser Sheikh, and Iain Matthews. Monocular object detection using 3d geometric primitives. In *the European Conference on Computer Vision (ECCV)*, pages 864–878, Berlin, Heidelberg, 2012.

- [CZM⁺19] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [CZP⁺18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *the European Conference on Computer Vision (ECCV)*, September 2018.
- [DBK⁺21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *the International Conference on Learning Representations (ICLR)*, 2021.
- [DBNVG17] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.
- [DDD09] Damien Delannay, Nicolas Danhier, and Christophe De Vleeschouwer. Detection and recognition of sports(wo)men from multiple views. In *2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–7, Aug 2009.
- [DKI⁺19] Adrien Delière, Ashwani Kumar, **Maxime Istasse**, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. Ordinal pooling. In *the British Machine Vision Conference (BMVC)*, pages 1–12, Cardiff, Wales, September 2019.
- [DLLT21] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems*, 2021.
- [DLS⁺09] Tiziana D’Orazio, Marco Leo, Paolo Spagnolo, Pier Luigi Mazzeo, Nicola Mosca, Massimiliano Nitti, and Arcangelo Disante. An investigation into the feasibility of real-time soc-

- cer offside detection from a multiple camera system. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(12):1804–1818, 2009.
- [DMS18] Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *the European Conference on Computer Vision (ECCV)*, pages 364–380, 2018.
- [EEVG⁺15] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.
- [FSW⁺19] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *the IEEE International Conference on Computer Vision (ICCV)*, October 2019.
- [FWR⁺17] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.
- [GCK⁺21] Remi Gribonval, Antoine Chatalic, Nicolas Keriven, Vincent Schellekens, Laurent Jacques, and Philip Schniter. Sketching data sets for large-scale learning: Keeping only what you need. *IEEE Signal Processing Magazine*, 38(5):12–36, 2021.
- [GCS⁺21] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2917–2927, 2021.
- [GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

- [GLM17] Sebastian Gerke, Antje Linnemann, and Karsten Müller. Soccer player recognition using spatial constellation features and jersey number recognition. *Computer Vision and Image Understanding (CVIU)*, 159:105 – 115, 2017.
- [GZI⁺21] Seyed Abolfazl Ghasemzadeh, Gabriel Van Zandycke, **Maxime Istasse**, Niels Sayez, Amirafshar Moshtaghpour, and Christophe De Vleeschouwer. Deepsportlab: a unified framework for ball detection, player instance segmentation and pose estimation in team sports scenes. In *the British Machine Vision Conference (BMVC)*, page 379, 2021.
- [HGD19] Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking imagenet pre-training. In *the IEEE International Conference on Computer Vision (ICCV)*, October 2019.
- [HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.
- [HPSL18] Jennifer Hobbs, Paul Power, Long Sha, and Patrick Lucey. Quantifying the value of transitions in soccer via spatiotemporal trajectory clustering. In *MIT Sloan Sports Analytics Conference*, 2018.
- [HSW90] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.
- [HysW⁺22] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *the IEEE*

- Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [IMDV19] **Maxime Istasse**, Julien Moreau, and Christophe De Vleeschouwer. Associative embedding for team discrimination. In *the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2477–2486, 2019.
- [IMV21] **Maxime Istasse**, Kim Mens, and Christophe De Vleeschouwer. ModulOM: Disseminating deep learning research with modular output mathematics. In *the International Conference on Learning Representations (ICLR), Beyond Static Papers Workshop*, 2021.
- [IMV23] **Maxime Istasse**, Amirafshar Moshtaghpour, and Christophe De Vleeschouwer. Resolving Occlusions when Segmenting Instances with Spatial Embeddings. 2023.
- [IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *the International Conference on Machine Learning (ICML)*, pages 448–456. pmlr, 2015.
- [Ist21] **Maxime Istasse**. “modulom over panoptic-deeplab” source code repository. <https://github.com/mistasse/modulom-panopticdeeplab>, 2021.
- [IV23] **Maxime Istasse** and Christophe De Vleeschouwer. Spatio-Associative Embeddings for Instance Segmentation in Complex Scenes. 2023.
- [JSH19] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.
- [JZD⁺23] Rohit Jena, Lukas Zhornyyak, Nehal Doiphode, Pratik Chaudhari, Vivek Buch, James Gee, and Jianbo Shi. Beyond map: Towards better evaluation of instance segmentation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

- [KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. [arXiv:1412.6980](#).
- [KBA19] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pif-paf: Composite fields for human pose estimation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [KDV13] KC Kumar and Christophe De Vleeschouwer. Discriminative label propagation for multi-object tracking with sporadic appearance features. In *the IEEE International Conference on Computer Vision (ICCV)*, pages 2000–2007, 2013.
- [KF18] Shu Kong and Charless C. Fowlkes. Recurrent pixel embedding for instance grouping. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9018–9028, 2018.
- [KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [KHG⁺19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [KL20] Victor Kulikov and Victor S. Lempitsky. Instance segmentation of biological images using harmonic embeddings. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3842–3850, 2020.
- [KMR⁺23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.
- [KPE21] Maria Koshkina, Hemanth Pidaparthi, and James H. Elder. Contrastive learning for sports video: Unsupervised player classification. In *the IEEE Conference on Computer*

- Vision and Pattern Recognition Workshops (CVPRW)*, *CVs-ports*, pages 4528–4536, June 2021.
- [KTT21] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. pages 4018–4027, Nashville, TN, USA, 2021.
- [KYL18] Victor Kulikov, Victor Yurchenko, and Victor S. Lempit-sky. Instance segmentation by deep coloring. *CoRR*, abs/1807.10007, 2018. [arXiv:1807.10007](#).
- [LAE⁺16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *the European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.
- [LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LC14] Jingchen Liu and Peter Carr. *Detecting and Tracking Sports Players with Random Forests and Context-Conditioned Motion Models*, pages 113–132. Cham, 2014.
- [LCLH18] Keyu Lu, Jianhui Chen, James J. Little, and Hangen He. Lightweight convolutional neural networks for player detection and classification. *Computer Vision and Image Understanding (CVIU)*, 172:77 – 87, 2018.
- [LD18] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *the European Conference on Computer Vision (ECCV)*, September 2018.
- [LEB08] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 1, page 3, 2008.
- [LGG⁺17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. pages 2999–3007, Venice, Italy, 2017.

- [LLM⁺18] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9628–9639, 2018.
- [LLW⁺17] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2978–2991, 2017.
- [LM16] Ke Li and Jitendra Malik. Amodal instance segmentation. In *the European Conference on Computer Vision (ECCV)*, pages 677–693, Cham, 2016.
- [LMB⁺14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *the European Conference on Computer Vision (ECCV)*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755, 2014.
- [LSC12] Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9(7):637–637, 2012.
- [LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
- [LTLM13] Wei-Lwun Lu, Jo-Anne Ting, James J Little, and Kevin P Murphy. Learning to track and identify players from broadcast sports videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1704–1716, 2013.
- [LWR⁺22] Peng Liu, Lizhe Wang, Rajiv Ranjan, Guojin He, and Lei Zhao. A survey on active deep learning: From model driven to data driven. *ACM Computing Surveys*, 54(10s), 2022.

- [LXZ⁺23] Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui, Yuning Du, Qingqing Dang, and Yi Liu. Detrs beat yolos on real-time object detection, 2023. [arXiv:2304.08069](https://arxiv.org/abs/2304.08069).
- [Maz18] Davide Mazzini. Guided upsampling network for real-time semantic segmentation. In *the British Machine Vision Conference (BMVC)*, September 2018.
- [MEAM17] M. Manafifard, H. Ebadi, and H. Abrishami Moghaddam. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding (CVIU)*, 159:19 – 46, 2017.
- [NALV18] David Novotný, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Semi-convolutional operators for instance segmentation. In *the European Conference on Computer Vision (ECCV)*, volume 11205 of *Lecture Notes in Computer Science*, pages 89–105, 2018.
- [NBPG19] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8837–8845, 2019.
- [ND17] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *Advances in Neural Information Processing Systems*, pages 2171–2180. 2017.
- [NHD17] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.
- [NORBK17] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kotschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *the IEEE International Conference on Computer Vision (ICCV)*, pages 4990–4999, 2017.
- [PBLZ18] Rudra P. K. Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. ContextNet: exploring context and detail

- for semantic segmentation in real-time. In *the British Machine Vision Conference (BMVC)*, September 2018.
- [PCKC16] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. June 2016. [arXiv:1606.02147v1](#).
- [PDV16] Pascaline Parisot and Christophe De Vleeschouwer. Consensus-based trajectory estimation for ball detection in calibrated cameras systems. *Journal of Real-Time Image Processing*, Sep 2016.
- [PDV17] Pascaline Parisot and Christophe De Vleeschouwer. Scene-specific classifier for effective and efficient team sport players detection from a single calibrated camera. *Computer Vision and Image Understanding (CVIU)*, 159:74 – 88, 2017.
- [PGM⁺19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.
- [PVG⁺11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [PZC⁺18] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *the European Conference on Computer Vision (ECCV)*, September 2018.
- [RDS⁺15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and

- Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.
- [RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015.
- [RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.
- [RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [SDVA23] Vladimir Somers, Christophe De Vleeschouwer, and Alexandre Alahi. Body part-based representation learning for occluded person re-identification. In *the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1613–1623, 2023.
- [SGG16] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 761–769, 2016.
- [Sha17] Meet Shah. Semantic segmentation architectures implemented in pytorch. <https://github.com/meetshah1995/pytorch-semseg>, 2017.
- [SHK⁺14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [SMFT14] Hanno Scharr, Massimo Minervini, Andreas Fischbach, and Sotirios A Tsaftaris. Annotated image datasets of rosette plants. In *the European Conference on Computer Vision (ECCV)*, pages 6–12, 2014.

- [TGM⁺17] Graham Thomas, Rikke Gade, Thomas B. Moeslund, Peter Carr, and Adrian Hilton. Computer vision for sports: Current applications and research topics. *Computer Vision and Image Understanding (CVIU)*, 159:3 – 18, 2017.
- [TLWZ11] Xiaofeng Tong, Jia Liu, Tao Wang, and Yimin Zhang. Automatic player labeling, tracking and field registration and trajectory mapping in broadcast soccer video. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(2):15:1–15:32, February 2011.
- [UCFB16] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.
- [URF⁺18] Jonas Uhrig, Eike Rehder, Björn Fröhlich, Uwe Franke, and Thomas Brox. Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In *IEEE Intelligent Vehicles Symposium (IV)*, 2018.
- [VSF⁺18] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *the European Conference on Computer Vision (ECCV)*, pages 402–419, Cham, 2018.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [VZDV19] Gabriel Van Zandycke and Christophe De Vleeschouwer. Real-time cnn-based segmentation architecture for ball detection in a single view setup. In *International Workshop on Multimedia Content Analysis in Sports*, pages 51–58, 2019.
- [WSC⁺19] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

- [WSL⁺15] Xinyu Wei, Long Sha, Patrick Lucey, Peter Carr, Sridha Sridharan, and Iain Matthews. Predicting ball ownership in basketball from a monocular view using only player trajectories. In *the IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.
- [WSvdH17] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Real-time semantic image segmentation via spatial sparsity. *CoRR*, abs/1712.00213, 2017. [arXiv:1712.00213](#).
- [WSVDH19] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.
- [YB15] Jiaqian Yu and Matthew Blaschko. Learning submodular losses with the lovász hinge. In *the International Conference on Machine Learning (ICML)*, pages 1623–1631. PMLR, 2015.
- [YCL16] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [YCZ⁺19] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papan-dreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv preprint arXiv:1902.05093*, 2019.
- [YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *the International Conference on Learning Representations (ICLR)*, 2016.
- [YWP⁺18] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: bilateral segmentation network for real-time semantic segmentation. In *the European Conference on Computer Vision (ECCV)*, September 2018.
- [ZQS⁺18] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for real-time semantic segmentation on high-resolution images. In *the European Conference on Computer Vision (ECCV)*, September 2018.

- [ZSI⁺22] Gabriel Van Zandycke, Vladimir Somers, **Maxime Istasse**, Carlo Del Don, and Davide Zambrano. Deepsportradar-v1: Computer vision dataset for sports understanding with high quality annotations. In *the ACM International Conference on Multimedia, MMSports Workshop*, pages 1–8, 2022.
- [ZSQ⁺17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [ZYH16] Stephan Zheng, Yisong Yue, and Jennifer Hobbs. Generating long-term trajectories using deep hierarchical networks. In *Advances in Neural Information Processing Systems*, pages 1543–1551, 2016.