Instance Segmentation of Dense Objects via Deep Pixel Embedding

Master Thesis

presented by

Yuli Wu

Supervisor:
Long Chen, M.Sc.

Institute of Imaging & Computer Vision
Prof. Dr.-Ing. Dorit Merhof
RWTH Aachen University

Erklärung nach §19 Abs. 7 DPO 2004

Aachen, 20.06.2020

Contents

List of Figures **VII**

List of Tables **VIII**

List of Listings **VIII**

Nomenclature **IX**

1 Introduction **1**

2 Related Work **5**

2.1 Convolutional Neural Networks 5

2.1.1 U-Net 6

2.1.2 Hourglass 6

2.1.3 ResNet + FPN 7

2.2 Loss Functions 8

2.2.1 Metrics of Distance 8

2.2.2 Cartesian Form of Embedding Loss 9

2.2.3 Polar Form of Embedding Loss 10

2.2.4 Circle Loss 10

2.3 Clustering Techniques 14

2.3.1 Radial and Angular Clustering 14

2.3.2 Mean Shift 15

2.3.3 Density Based Clustering 17

2.3.4 Mutex Watershed 18

2.4 State of the Art 19

2.4.1 Instance-First Approaches 19

2.4.2 One-Stage Approaches 21

2.4.3 Deep Embedding Learning Approaches 21

2.4.4 Other Approaches on Leaf Segmentation 22

2.5 Other Techniques 23

3 Method **27**

3.1 Processing Pipeline 28

3.2 Cosine Embedding Loss with Local Constraints 29

3.3 Feature Concatenative Layer 30

3.4 From U-Net to W-Net 31

4 Experiments **33**
4.1 Datasets . . . 33
4.1.1 CVPPP Leaf Segmentation . . . 34
4.1.2 BBBC006: U2OS Cells . . . 34
4.1.3 Cityscapes . . . 35
4.2 Ablation Experiments . . . 36
4.2.1 Cartesian vs. Polar Embedding . . . 36
4.2.2 Local vs. Global Constraints . . . 38
4.2.3 U-Net vs. W-Net . . . 41
4.2.4 Concatenative Layer . . . 45
4.2.5 Loss Weights as Hyper-Parameters . . . 47
4.2.6 Adaptive vs. Constant Loss Weights . . . 49
4.2.7 Dimension of Embeddings . . . 53
4.2.8 Loss for Distance Regression . . . 55
4.2.9 Clustering . . . 56
4.3 Comparison against State-of-the-Art . . . 57
4.4 Application on Human U2OS Cells . . . 58
4.4.1 Results . . . 58
4.5 Application on Cityscapes . . . 60
4.5.1 Augmentation . . . 60
4.5.2 Visual Results . . . 61

5 Evaluation **65**
5.1 Weaknesses . . . 65
5.1.1 Demanding Training Images . . . 66
5.1.2 Scale Inflexibility . . . 67
5.1.3 A Detour for Accuracy . . . 67
5.1.4 Application Constraints . . . 68
5.2 Rethinking Similarity Loss Pair . . . 68

6 Final Remarks **71**
6.1 Conclusions . . . 71
6.1.1 Concatenative Layer . . . 72
6.1.2 Pixel Embedding Learning . . . 72
6.2 Outlooks . . . 73
6.2.1 End-to-End Multi-class Instance Segmentation . . . 73
6.2.2 Similarity Loss Pair . . . 73

Bibliography **i**

A Network Architectures **ix**

B Extension: Ensemble Trick **xi**

C Online Access **xiii**

List of Figures

1.1 Structure of this Thesis 2

2.1 U-Net 6
2.2 Stacked Hourglass 7
2.3 Feature Pyramid Networks 7
2.4 Optimization of Similarity Pair with Adaptive Weights 11
2.5 Toy Function $y = x + \log(e^{-x} + e^{-a})$ 13
2.6 Adaptive Weighting Factors of Circle Loss 14
2.7 Radial and Angular Clustering 15
2.8 Mask R-CNN 20
2.9 Hourglass with Cosine Embeddings 22
2.10 Example of Distance Transform 23
2.11 Activation Functions 24

3.1 Ambiguity between Leaf Boundary and Leaf Midvein 27
3.2 Processing Pipeline 28
3.3 Learned Embeddings with U-Net and W-Net 30
3.4 Hybrid Network Architectures of U-Net and W-Net 31

4.1 Examples of Leaf Segmentation 34
4.2 An Example of BBBC006: U2OS Cells 35
4.3 An Example of Cityscapes 36
4.4 Examples of Euclidean Embeddings with Global Constraints 37
4.5 Joint Block between Backbone and Loss 38
4.6 Learned Embeddings of Different Constraints and Embedding Dimensions 40
4.7 U-Net and Variants: Simplified 42
4.8 Leaf Segmentation Results of the CVPPP2017 Testing Set 44
4.9 Distmap as Concatenative Layer 45
4.10 Results of Different Loss Weights 48
4.11 Learned Embeddings with Different Weight 49
4.12 Comparison of Constant Loss Weights 50
4.13 An Example of $\mathcal{K}(\omega)$ 51
4.14 Comparison of Adaptive Weights 52
4.15 Comparison of Adaptive Weights 53
4.16 Results of Different Embedding Dimensions 54
4.17 Loss Comparison for Distance Regression 55
4.18 Cell Segmentation Results of the BBBC006v1 Data 59
4.19 Augmentation for Cityscapes 60

4.20 Citycapes Results Part 1 62
4.21 Citycapes Results Part 2 63

5.1 Failed Cases 66
5.2 Expected Processing Pipeline towards Panoptic 67
5.3 Dilemma of Embeddings 68

6.1 Outlook of Embeddings 72

A.1 Original U-Net. 2.1 revisited. ix
A.2 U-Net with 2 Heads. ix
A.3 Stacked U-Net (W-Net). x
A.4 Stacked U-Net (W-Net) with Intermediate Supervision. x

B.1 Leaf Segmentation Results of the CVPPP2017 Testing Set A3 xii

List of Tables

2.1 Comparison of Deep Embedding Learning Methods 21

4.1 Roles of 3 Datasets in this Thesis 33
4.2 Four Formats of Loss Functions. 39
4.3 Comparison of U-Net Variants 43
4.4 Comparison of Different Types of Concatenative Layers 47
4.5 Results of Adaptive and Constant Loss Weights 52
4.6 Comparison of Local/Global Constraints, Network and Clustering . . 57
4.7 Comparison against State-of-the-Art 58

5.1 Results of Testing Sets 65

List of Listings

2.1 Pseudocode of DBSCAN Algorithm 16
2.2 Pseudocode of Mutex Watershed 18

Nomenclature

$\mathcal{F}(\boldsymbol{x})$	tensor as network output
$\boldsymbol{x}$	tensor as network input
$\mathcal{G}_n(\boldsymbol{x})$	tensor as output of n-th layer
$[\boldsymbol{x}_1, \boldsymbol{x}_2]$	concatenation of two tensors
D_{euc}	Euclidean distance
D_{cos}	cosine distance
D_{ang}	angular distance
S_{euc}	Euclidean similarity
S_{cos}	cosine similarity
$\boldsymbol{e}$	embedding vector
$\|\|\boldsymbol{e}\|\|_2^2$	L2 norm of vector $\boldsymbol{e}$
$\boldsymbol{e}^T$	transposed vector
$[a]_+$	$\max\{a, 0\}$
$\mathcal{L}_{emb}$	loss function of embedding
$\mathcal{L}_{intra}$	between-instance loss term
$\mathcal{L}_{inter}$	within-instance loss term
C	number of instances
c_A	object with label A
$\boldsymbol{\mu_c}$	centroid embedding of object c
δ, Δ	margin
E_c	number of pixels in object c
$\mathbf{N}_c$	set of neighboring objects of object c
$\|\mathbf{N}_c\|$	number of neighboring objects of object c
λ	weighting factor before $\mathcal{L}_{intra}$
γ	scale factor
α	adaptive weighting factor
$\hat{\mathcal{L}}$	loss value as a constant
$\mathcal{K}(\omega)$	adaptive weighting factor as a function

1 Introduction

Instance segmentation aims to label each individual object, which is critical to many biological and medical applications, such as plant phenotyping and cell quantification, and also to many industrial applications and even daily life, such as the vision-based self-driving algorithm.

Before entering the technical parts, the importance of the mentioned fields is introduced. Plant phenotyping is an emerging science about the identification of effects on plant structure and function (the phenotype), that links genomics with plant ecophysiology and agronomy [1]. It can provide essential information on how differences in genetic code, environmental conditions which a plant has been exposed to, and crop management can guide selection toward productive plants suitable for their environment to ensure, for example, global food security. [2, 3] The most primarily used dataset in this thesis is the Computer Vision Problems in Plant Phenotyping (CVPPP) Leaf Segmentation Challenge (LSC) of the latest version in 2017 [4], which contains the images of Arabidopsis and Tobacco. Arabidopsis is considered as a first class model organism and the single most important species for fundamental research in plant molecular genetics [5]. Moreover, many discoveries with direct relevance to human health and disease have been elaborated using Arabidopsis [6]. The proposed method has also been tested on the image set BBBC006v1 of human U2OS cells from the Broad Bioimage Benchmark Collection [7]. The human osteosarcoma (bone cancer) U2OS cell line is one of the first generated cell lines and is used in various areas of biomedical research [8]. Last but not least, the widely used semantic understanding of urban street scenes dataset Cityscapes [9] has also been tested in this work, which shows the promising application on the towards real-world scenarios. This thesis is not only inspired by the trending area of computer vision and machine learning (particularly deep learning), but also motivated by the desirable techniques in the biomedical research and even daily life.

Learning object-aware pixel embeddings is one of the trends in the field of instance segmentation. The embedding is essentially a high-dimensional representation of each pixel. To achieve instance segmentation, pixel embeddings of the same object should be located relatively close in the learned embedding space, while those of different objects should be discriminable. The loss usually consists of two terms: the between-instance loss term $\mathcal{L}_{inter}$ and the within-instance loss term $\mathcal{L}_{intra}$. The former term $\mathcal{L}_{inter}$ encourages different-instance embeddings to be located far away from each other, while the latter term $\mathcal{L}_{intra}$ encourages same-instance embeddings to stay together. Two most popular metrics used to describe the similarities of embeddings are Euclidean distance and cosine distance. Although the pixel embedding

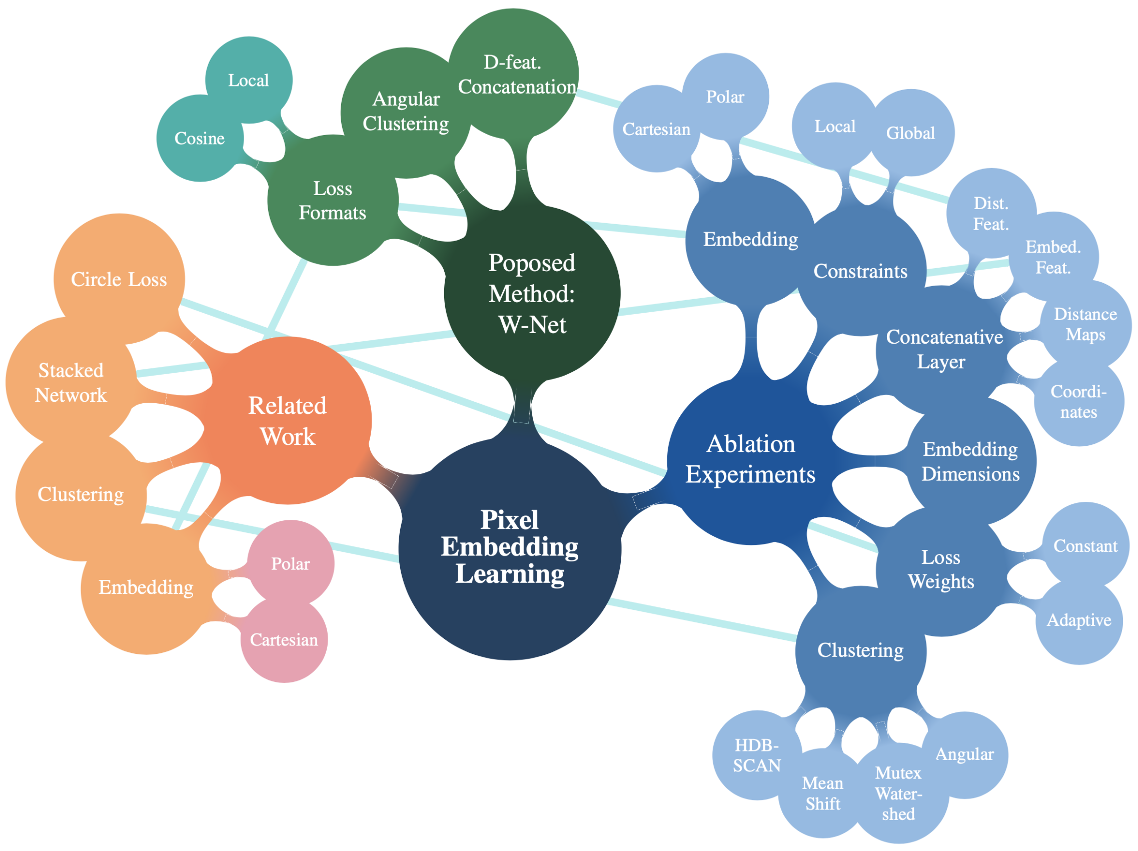


Figure 1.1: Structure of this Thesis.
Chapter 2 - Chapter 4 are illustrated as orange, green and blue, respectively. Connecting lines denote the inspiration or corroboration.

approaches have gained success in many datasets including CVPPP Leaf Segmentation Challenge [10–13], the trained embedding space is far from optimal.

The idea was indirectly inspired by the “easy task first” concept behind curriculum learning [14]. Distance regression predicts the distance from a pixel to the object boundary and is used in [10, 15], for example, as an auxiliary module. Empirically, it is found that the distance regression module is relatively easy to train on many datasets. Considering that the learned features by the distance regression module should be already recognizable for distinguishing instances, the embedding module is prefixed with a distance regression module to promote the embedding learning process.

This thesis is structured as follows. Chapter 2 covers the related work in the field of instance segmentation, including the state-of-the-art algorithms using instance-first strategy and one-stage approaches. In particular, the methods based on the pixel embedding learning are introduced and compared. Following that, the proposed method is demonstrated in Chapter 3, which results in the best performance throughout the ablation experiments. Next, the detailed ablation experiments are introduced in Chapter 4, covering the network architectures, the loss formats and the clustering techniques. Based on the experimental results, some questions are raised in Chapter 5, which are worth further research in the future. At last, the conclusions are summarized and discussed in Chapter 6. Figure 1.1 gives an overview of the structure in this thesis.

The main contributions of this thesis are summarized as follows:

1. A novel architecture has been proposed to promote the pixel embedding learning by utilizing features learned from the distance regression module, which significantly improves the performance in the CVPPP Leaf Segmentation Challenge [4]. Our overall mean Symmetric Best Dice (mSBD) score is at the top position of the leaderboard with 0.879 by thesis submission. Furthermore, the average of mSBD scores on Arabidopsis images (testing sets A1, A2, A4) outperforms the second best results from three different teams by over 3%, namely from 0.883 to 0.917;

2. A number of ablation experiments have been conducted in terms of the stacked U-Net architecture, different types of concatenative layers and varied loss formats, to validate our architecture and also supplement some experimental vacancies in this field;

3. The proposed method has also been tested on Cell Segmentation and Cityscapes and the improved capability has been confirmed;

4. Problems of similarity loss pair have been raised and the promising future work is itemized.

2 Related Work

In addition to Mask R-CNN, one of the paradigms of instance segmentation algorithms, a considerable number of approaches are made, especially those via deep embedding learning. The fundamental components in the deep embedding learning pipeline are as follows: a convolutional neural network, served as feature extractor which map pixels from original images to multi-dimensional vectors into embedding domain, a loss function, served as objective function in the optimization problem with the learned features from the convolutional neural network being the input elements, and a clustering block, served as post-processing which transforms embeddings to labels.

With this in mind, the convolutional neural network architectures, loss functions and clustering techniques are introduced in Sections 2.1 - 2.3. Chosen state-of-the-art algorithms are demonstrated and preliminarily compared in Section 2.4, which covers three categories: instance-first approaches, one-stage approaches and deep embedding learning approaches. The grouping is not meant to be mutually exclusive, but to involve the most appealing trends, including those from CVPR 2020. Following that, other algorithms and techniques, which are relevant in the experiments, are introduced to fill the gap of following chapters in Section 2.5.

2.1 Convolutional Neural Networks

It is hardly possible in the recent years to talk about image processing and computer vision without mentioning convolutional neural networks (CNN). No exception that the CNN is used in this thesis as a feature extractor. Three popular models are introduced, which share one same characteristic: they exploit the multiple scales of feature maps. The combination of multi-scales and deep layers tend to be the must-have characters of modern architectures.

The chosen network architecture used in the experiments as backbone under the coverage of this thesis is U-Net due to its fundamental yet sufficient components. The more novel architectures may be advantageous, but it is out of the scope of this thesis. Despite that, the differences of few popular networks are worth being briefly introduced to indicate how the final modified U-Net is inspired. Future work containing the comparison of different backbone architectures in the field of instance segmentation via deep embedding learning are promising and expected.

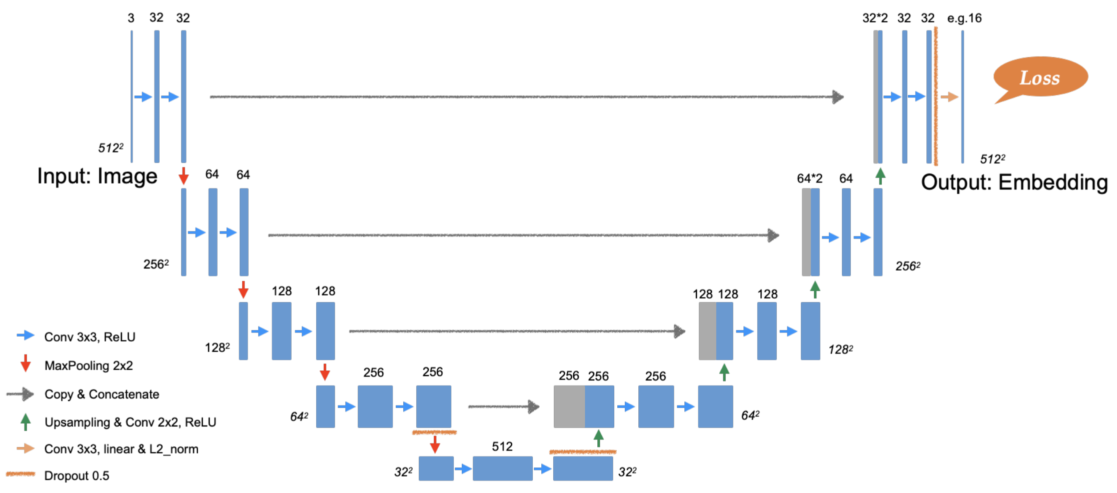


Figure 2.1: U-Net Architecture

2.1.1 U-Net

Focused on the biomedical images, U-Net [16] has established the foundations and principles of upcoming CNNs. The two salient characteristics of U-Net are as follows, which can be noticed in Figure 2.1:

- Symmetric down-sampling and up-sampling exploit different levels context density of feature maps: the lower, the more dense contexts are;
- Skip connections fuse information from different abstract levels: the later, the more abstract feature maps are.

[17] has proposed a modified and more complicated architecture: U-Net++, where more connections between low and high blocks are designed. [18] represented a novel dense connectivity, termed dense block, which shares the same idea of the usage of concatenation. Thanks to the densely connected architecture, the supervision of the loss function is more direct and thus more effective than the previously proposed networks. And the collective knowledges are more effectively shared. Because it exploits concatenation as skip connection, which is the case of U-Net, it is introduced in this section. Meanwhile, the complete network, termed *DenseNet*, is usually regarded to as a variant of *ResNet*, which is introduced in Section 2.1.3.

2.1.2 Hourglass

[19] proposed an hourglass architecture originally for human pose estimation, where the vanilla networks are stacked to each other to improve the performance as illustrated in Figure 2.2. It is noticeable that in [19] the effectiveness of *Intermediate Supervision* has been demonstrated. The performance is improved, if the feature maps between the stacked networks are also added by loss functions, which is termed as *Intermediate Supervision*. Stacked networks extend the potential of arbitrary architectures further, as it shows the possibility of improvement by stacking and

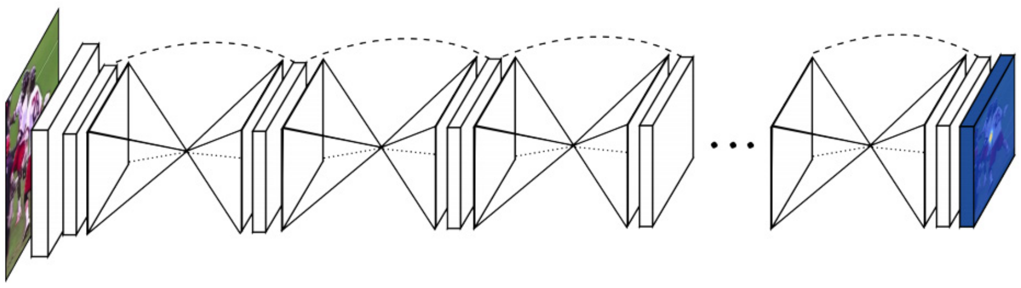


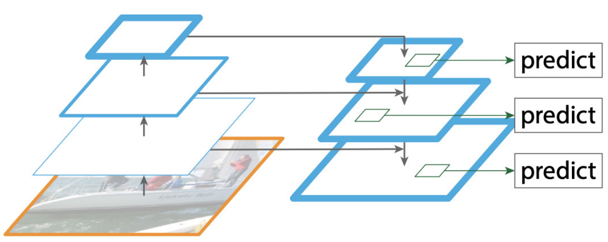
Figure 2.2: Stacked Hourglass Architecture. Adapted from [19].

intermediately supervising the original architectures. In contrast, the saturation of the redundant stacked networks and the trade-off between overhead for computation and memory and the relative improvement is worth being investigated.

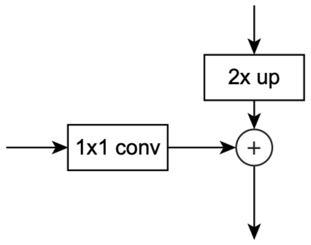
2.1.3 ResNet + FPN

Before [20], the very deep networks (more than 100 layers) usually suffered from inefficiency. Thanks to the novel skip connections, the trade-off of depth of network and efficiency has been mitigated.

As mentioned previously, the DenseNet is considered as a variant of ResNet. The major difference is the type of skip connection, where the dense concatenation is used in DenseNet and addition in ResNet. As a result, the output of a *Dense Block* is defined as $\mathcal{F}(\boldsymbol{x}) = \mathcal{G}_n([\,\mathcal{G}_{n-1}(\boldsymbol{x}), \mathcal{G}_{n-2}(\boldsymbol{x}), ..., \boldsymbol{x}\,])$, while the output of a *ResNet Block* is defined as $\mathcal{F}(\boldsymbol{x}) = \mathcal{G}_n(\boldsymbol{x}) + \boldsymbol{x}$, where $\mathcal{F}(\boldsymbol{x})$ denotes the output of block with the input $\boldsymbol{x}$, $\mathcal{G}(\boldsymbol{x})$ denotes the output of previous layer in this block, and $[\,..,..\,]$ denotes the concatenation operation. Other variants of ResNet are, for instance, ResNeXt [21] and ResNeSt [22].



(a) FPN



(b) Lateral Connection

Figure 2.3: Feature Pyramid Networks.
(b): Lateral connection between the top-down pathway and skip connection, merged by addition.
Adapted from [23]

ResNet is a deep yet serial-shaped architecture, which lacks the ability of multi-scaling. Thus the combination of ResNet and Feature Pyramid Networks (FPN) [23] becomes the new trend in the field of object detection, such as in Fast R-CNN and its successors. As the name implies, the pyramid shape is the main characteristic to extract features of multi-scales. The lateral connection collects information from high-resolution (output of `1x1 conv`) and high-context (output of `2x up`) via addition, as illustrated in Figure 2.3. Notice that the upsampling is used to hallucinate high-resolution features, which is the same as U-Net. Yet the additive connection is different from the concatenation used in U-Net.

2.2 Loss Functions

In this section, a set of loss functions are introduced. The construction of loss functions plays a critical role in deep learning, as they connect the learned features from the outputs of deep learning network with the final evaluation metrics, and thus they determine, in which manner the target problem should be optimized.

In the field of instance segmentation via embedding learning, one of the most intuitive formats of building loss functions is the joint loss consisting of two terms: the *between-instance loss* and the *within-instance loss.* In this thesis, they are also depicted as inter loss ($\mathcal{L}_{inter}$) and intra loss ($\mathcal{L}_{intra}$). Since the losses are depicted through the similarity pair, *between-instance similarity* (s_n) and the *within-instance similarity* (s_p), these two terms are also used in this thesis to describe the non-weighted versions. With context it is clear that the optima of $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ are 0, and of s_n and s_p are 0 and 1 respectively.

The metrics of distance, Euclidean Distance and Cosine Distance, are introduced in Section 2.2.1. Based on that, the chosen embedding loss functions are categorized into two types: Cartesian Form and Polar Form, presented from [11] and [10] respectively. In Section 2.2.4, Circle Loss is introduced together with its degenerated variants AM-Softmax and Triplet Loss, where Circle Loss provides a novel flexible optimization manner of similarity pair.

2.2.1 Metrics of Distance

To describe the similarity and distance (namely dissimilarity) of two embeddings, two metrics are demonstrated: Euclidean Distance (or referred to as L2 Norm Distance), and Cosine Distance. The criterion of these metrics of distance is that their measures should be non-negative and the larger the measures are, the more distinct the two embeddings should be.

They are defined as below:

Euclidean Distance

$$D_{euc}(\mathbf{e_1}, \mathbf{e_2}) = 1 - S_{euc}(\mathbf{e_1}, \mathbf{e_2}) = \|\mathbf{e_1} - \mathbf{e_2}\|_2^2 \tag{2.1}$$

Cosine Distance

$$D_{cos}(\mathbf{e_1}, \mathbf{e_2}) = 1 - S_{cos}(\mathbf{e_1}, \mathbf{e_2}) = 1 - \frac{\mathbf{e_1}^T \cdot \mathbf{e_2}}{\|\mathbf{e_1}\|_2^2 \cdot \|\mathbf{e_2}\|_2^2} \tag{2.2}$$

One salient difference between these two formats is, that the measures of Euclidean Distance can be arbitrarily large, whereas the measures of Cosine Distance are ranged between 0 and 1. Speaking of Cosine Distance, the measures are 1 if the two embeddings are orthogonal, or 0 if the two embeddings are identical. And vice versa when speaking of Cosine Similarity (denoted as S_{cos}).

One thing worth mentioning is that the Cosine Distance violates the triangle inequality. One counterexample can be easily found: Let $\boldsymbol{a} = [1, 0], \boldsymbol{b} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}], \boldsymbol{c} = [0, 1]$. It results in $D_{cos}(\boldsymbol{a}, \boldsymbol{c}) = 1, D_{cos}(\boldsymbol{a}, \boldsymbol{b}) = D_{cos}(\boldsymbol{b}, \boldsymbol{c}) = 1 - \dfrac{1}{\sqrt{2}}$, which educes $D_{cos}(\boldsymbol{a}, \boldsymbol{c}) > D_{cos}(\boldsymbol{a}, \boldsymbol{b}) + D_{cos}(\boldsymbol{b}, \boldsymbol{c}) \approx 0.6$. The property of triangle inequality can be repaired by a modified definition of Angular Distance:

$$D_{ang}(\mathbf{e_1}, \mathbf{e_2}) = \frac{\arccos(S_{cos}(\mathbf{e_1}, \mathbf{e_2}))}{\pi}, \tag{2.3}$$

which is also bounded between 0 and 1.

Based on the different description of distance, corresponding embedding loss functions can be defined in two forms, which are introduced in the following two sections.

2.2.2 Cartesian Form of Embedding Loss

The Cartesian Form of Embedding Loss uses the Euclidean distance. Equation 2.4 showcases one example proposed in [11], where the distances with margins are squared.

$$\begin{aligned}
\mathcal{L}_{emb} &= \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C(C-1)} \sum_{c_A=1}^{C} \sum_{\substack{c_B=1 \\ c_A \neq c_B}}^{C} \Big[D_{euc}(\boldsymbol{\mu_{c_A}}, \boldsymbol{\mu_{c_B}}) - 2\delta_1 \Big]_+^2 \\
\mathcal{L}_{intra} &= \frac{1}{C} \sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[\delta_2 - D_{euc}(\boldsymbol{e_i}, \boldsymbol{\mu_c}) \Big]_+^2,
\end{aligned} \tag{2.4}$$

where C denotes the number of instances; c_A, c_B denote labels; $\boldsymbol{\mu_{c_A}}, \boldsymbol{\mu_{c_B}}, \boldsymbol{\mu_c}$ denote centers of embedding clusters labeled with c_A, c_B, c which are calculated as mean embeddings; e_i denotes the i-th embedding in the respective cluster c with totally E_c embeddings; δ_1, δ_2 denote margins; the operator $[\cdot\cdot]_+$ denotes clip-by-0 if negative values are given. An illustration can be found in Figure 2.7(a) together with the radial clustering technique.

2.2.3 Polar Form of Embedding Loss

Analogue to Cartesian Form, Polar Form of Embedding Loss is defined as follows:

$$\begin{aligned}
\mathcal{L}_{emb} &= \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C}\sum_{c_A=1}^{C} \frac{1}{|\mathbf{N}_{c_A}|} \sum_{c_B \in \mathbf{N}_{c_A}} \Big[1 - D_{cos}(\boldsymbol{\mu_{c_A}}\,,\, \boldsymbol{\mu_{c_B}}) \Big] \\
\mathcal{L}_{intra} &= \frac{1}{C}\sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[D_{cos}(\boldsymbol{e_i}\,,\, \boldsymbol{\mu_c}) \Big],
\end{aligned} \tag{2.5}$$

where $|\mathbf{N}_{c_A}|$ denotes the number of instances which are adjacent to instance labeled with c_A. Other notations can be referred to after Equation 2.4. This form is originally proposed in [10].

According to Four Color Theorem [25], only 4 labels are needed to segment adjacent objects sufficiently. Inspired by this, it would simplify the instance segmentation process by only considering the neighborhoods of target instance. Apart from the metrics of distance, the local constraint is another and salient distinction to [11]. Furthermore, the distance is not squared in contrast with the previously introduced Cartesian Form. In the original paper [10], no margins are added in the loss functions, which is also one difference to Cartesian Form. An illustration can be found in Figure 2.7(b) together with the angular clustering technique.

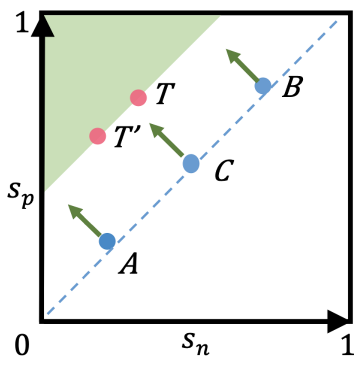
2.2.4 Circle Loss

It is challenging to optimize the *between-instance loss* and the *within-instance loss* simultaneously and stepwise equivalently, if they are additively jointed. In the field of semantic segmentation and facial recognition, [26] proposed a novel loss, which combines the similarity pair together with adaptive weights to avoid the extreme case that the optimization process is stuck as one term has reached its optimum, as illustrated in Figure 2.4. In this work, cosine similarity is used. In Section 2.2.4, the *between-instance* similarity is denoted as s_n and the *within-instance* similarity is denoted as s_p, indicating that their optima should be 0 (n for negative) or 1 (p for positive) respectively.

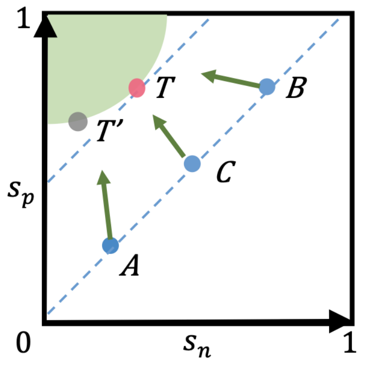
The Circle Loss is defined as:

$$\mathcal{L}_{circle} = \log\Big[1 + \sum_{j=1}^{L} \exp(\gamma \alpha_n^j (s_n^j - \Delta_n)) \sum_{i=1}^{K} \exp(-\gamma \alpha_p^i (s_p^i - \Delta_p)) \Big], \tag{2.6}$$

where L, K denote the number of the *between-instance* and the *within-instance* similarity pairs, γ denotes scale factor, α denotes adaptive weighting factor, Δ denotes margin.



(a) Vanilla Approach



(b) Circle Loss Approach

Figure 2.4: Optimization of Similarity Pair with Adaptive Weights.
(a): Combine similarity pair through $(s_n - s_p)$;
(b): Combine similarity pairs with adaptive weights $(\alpha_n s_n - \alpha_p s_p)$.
Take point A as an example: if s_n is much closer to its optimum (0) than s_p to its (1), using adaptive weights can avoid optimization process being stuck in the left boundary. In this example, the adaptive weights should be small for s_n and large for s_p respectively. As a result, the target T is more preferable than T'.
Adapted from [26].

The Circle Loss provides a unified perspective for *learning with class-level labels and with pair-wise labels.* Two representative loss functions which can be obtained with slight modifications are demonstrated: *AM-Softmax Loss* [27] for class-level labels and *Triplet Loss* [28] for pair-wise labels.

AM-Softmax Loss: First the adaptive weighting factors are neglected $\alpha_n = \alpha_p = 1$. Then the two margins are simplified as one additive margin $\Delta_{am} = \Delta_p - \Delta_n$, being added only to the between-class similarity pairs. Finally by setting the number of within-class similarity pairs K to 1, the AM-Softmax Loss [27] can be degenerated from Equation 2.6 as below:

$$\begin{aligned}\mathcal{L}_{AM-Softmax} &= \log\left[1 + \sum_{j=1}^{L-1} \exp(\gamma(s_n^j + \Delta_{am}))\exp(-\gamma s_p)\right] \\ &= \log\left[1 + \frac{\sum_{j=1}^{L-1}\exp(\gamma s_n^j)}{\exp(\gamma(s_p - \Delta_{am}))}\right] \\ &= -\log\left[\frac{\exp(\gamma(s_p - \Delta_{am}))}{\exp(\gamma(s_p - \Delta_{am})) + \sum_{j=1}^{L-1}\exp(\gamma s_n^j)}\right]\end{aligned} \tag{2.7}$$

Triplet Loss: With omitted adaptive weighting factors $\alpha_n = \alpha_p = 1$ from Equation 2.6, the Triplet Loss [28, 29] can be degenerated as below:

$$\begin{aligned}
\mathcal{L}_{Triplet} &= \lim_{\gamma\to+\infty} \frac{1}{\gamma}\mathcal{L}_{circle} \\
&= \lim_{\gamma\to+\infty} \frac{1}{\gamma}\log\Big[1+\sum_{i=1}^{K}\sum_{j=1}^{L}\exp(\gamma((s_n^j - s_p^i)-(\Delta_n-\Delta_p)))\Big] \\
&= \max[s_n^j - s_p^i]_+
\end{aligned} \tag{2.8}$$

Equation 2.8 indicates that one of the most distinguished differences between Circle Loss and vanilla approach $(s_n - s_p)$ is the *min-max* manner of optimization target suggested in Circle Loss. Without loss of generality it can be deduced with the following simplified *log-sum-exp* format:

$$\begin{aligned}
& y = \log(\,1+\sum_{n=1}^{N} e^{x_n}\,) \\
\Leftrightarrow\quad & e^y = 1+\sum_{n=1}^{N} e^{x_n} \\
\Leftrightarrow\quad & (e^y-1)\cdot e^{-\max\{x_n\}_{n=1}^N} = \sum_{n=1}^{N} e^{x_n-\max\{x_n\}_{n=1}^N} \\
\Leftrightarrow\quad & e^{y-\max\{x_n\}_{n=1}^N} = e^{-\max\{x_n\}_{n=1}^N} + \sum_{n=1}^{N} e^{x_n-\max\{x_n\}_{n=1}^N} \\
\Leftrightarrow\quad & y = \max\{x_n\}_{n=1}^N + \log(\,e^{-\max\{x_n\}_{n=1}^N} + \underbrace{\sum_{n=1}^{N} e^{x_n-\max\{x_n\}_{n=1}^N}}_{\leqslant N}\,)
\end{aligned} \tag{2.9}$$

To this end, the optimization problem of minimizing y can thus be incorporated to two sub-targets:

- to minimize the maximal value $\max\{x_n\}_{n=1}^N$, and
- to minimize distances between all $\{x_n\}_{n=1}^N$ and their maximal value.

The adaptive weighting factors α_n, α_p are designed in a self-paced manner as follows:

$$\begin{cases} \alpha_n^j = [s_n^j - O_n]_+ \\ \alpha_p^i = [O_p - s_p^i]_+\,, \end{cases} \tag{2.10}$$

where O_n, O_p denote optima of the *between-instance* and *within-instance* similarity pairs respectively.

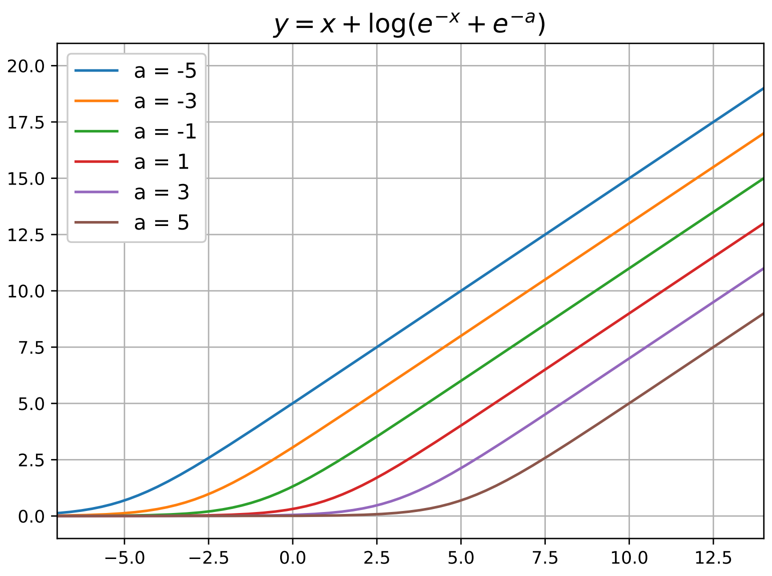


Figure 2.5: Plot of Toy Function $y = x + \log(e^{-x} + e^{-a})$ with Ascending a. With substitution of $\max\{x_n\}_{n=1}^N$ from Equation 2.9 by x, and of approximated underbraced term by e^{-a}.
It is clear that the minimal y is reachable with smaller x and greater a, indicating the two sub-targets of minimizing y: to minimize the maximal value $\max\{x_n\}_{n=1}^N$, and to minimize distances between all $\{x_n\}_{n=1}^N$ and their maximal value.

The following four hyper-parameters are further reduced through:

$$\begin{cases} O_n = -m \\ O_p = 1 + m \\ \Delta_n = m \\ \Delta_p = 1 - m\,, \end{cases} \tag{2.11}$$

where m denotes relaxation factor, which controls both adaptive weighting factors and margins. In the original paper, the relaxation factor m is empirically set as 0.25 to achieve best results. It indicates that the optima $O_n = -0.25, O_p = 1.25$, which are beyond the theoretical range of cosine similarity. The benefit is obvious, inasmuch as even if the between-class similarity is optimized ($s_n \to 0$), for example, it can still make a *minor yet non-negligible* contribution to the total loss with weighting factor $\alpha_n = 0.25$ to avoid losing dynamics.

To this end, only 2 hyper-parameters remain, namely the scale factor γ and the relaxation factor m. With substitutions from Equation 2.10 and Equation 2.11,

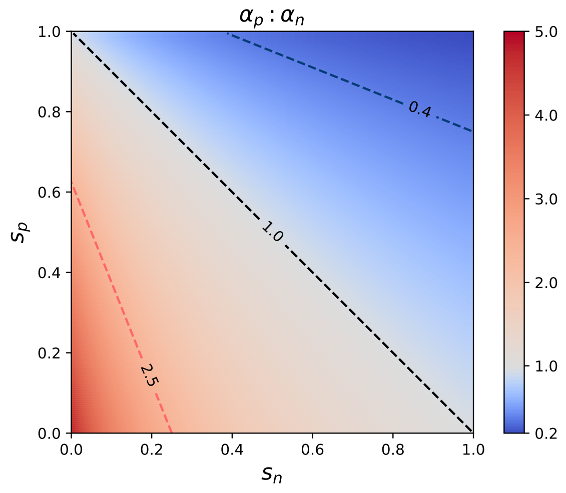


Figure 2.6: Adaptive Weighting Factors of Circle Loss.
Colors denote the ratio of two weighting factors $\alpha_p : \alpha_n$. Dashed lines illustrate three representative isolines, indicating for example the s_p term plays a more significant role in the optimization process than s_n when $\alpha_p : \alpha_n = 2.5 > 1$. The trend of isolines corresponds with Figure 2.4. Weighting factors are defined as referred to Equation 2.10 and Equation 2.11 with $m = 0.25$.

Equation 2.6 can thus be rewritten as below:

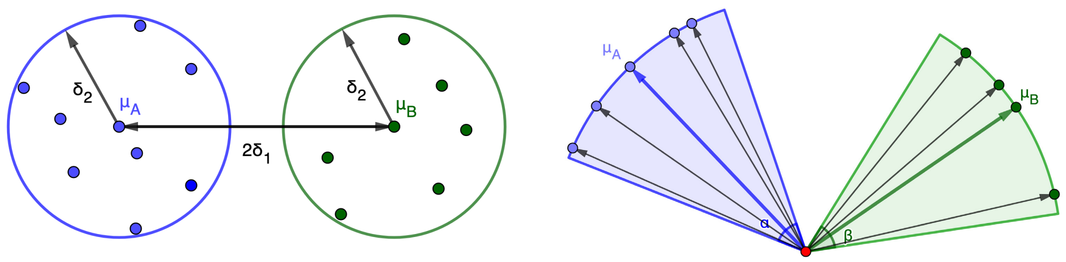
$$\mathcal{L}_{circle} = \log\Big[1 + \sum_{j=1}^{L} \exp(\gamma[s_n^j + m]_+(s_n^j - m)) \sum_{i=1}^{K} \exp(-\gamma[1 + m - s_p^i]_+(s_p^i - 1 + m))\Big] \tag{2.12}$$

To sum up, the Circle Loss provides a mechanism upon the similarity loss pair to regularize the numerical values of two loss terms. It is expected to avoid the case that the optimization processing is terminated as the one loss term is well converged while the other not. In the ablation experiments, the simplified version is investigated to build the adaptive version of loss weights. Although the results show that the overall performance has not been improved with the adaptive loss weighting factors, it is believed that the attempt is valuable and is promising in the further similarity loss pair based approaches.

2.3 Clustering Techniques

2.3.1 Radial and Angular Clustering

One of the most intuitive approaches of clustering is radial or angular clustering. Based on the preselected clustering centers (or so-called seeds) and predefined lengths or angles as thresholds, all embeddings within these areas are clustered as one instance, as shown in Figure 2.7.



(a) Radial Clustering **(b)** Angular Clustering

Figure 2.7: Radial and Angular Clustering.
μ_A, μ_B denote the centers of cluster A (blue) and B (green) respectively.

The difficulties of this understandable approach are:

- The seeds should be learned, which may decrease the accuracy of final clustering results.
- The thresholds should be learned, which may decrease the accuracy of final clustering results; or self-defined, which may be hard to choose especially for instances with varied forms and sizes.

It is advantageous in computation overhead and is geometrically meaningful, especially when margins between and within instances are applied. Figure 2.7 (a) shows that the hyper-parameter radius is related to the within-class margin δ_2 in Equation 2.4.

2.3.2 Mean Shift

Mean Shift [30–32] is a centroid based clustering method, where the number of the clusters is not required in advance. As this is the case in the field of instance segmentation, unnecessary parameter of the number of clusters is one of the criteria for the clustering techniques for instance segmentation. Since there are some variants of Mean Shift with different kernels or other implementation details, the following introduction is based on the one implemented in the Python package scikit-learn [33].

The Mean Shift works as follows. The to-be-shifted mean vector for the i-th seed of t-th iteration on the basis of $(t-1)$-th mean vector $m^{t-1}(x_i) = x_i$ is defined as $m^t(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$, where the Gaussian Radial Basis Function Kernel (RBF Kernel) is used $K(x_j - x_i) = \exp\left(-\frac{||x_j - x_i||^2}{2h^2}\right)$ with the bandwidth h, which also defines the grid size for sampling in the initial stage and the threshold for defining neighborhoods.

Ostensibly, one of the advantages of Mean Shift is the single hyper-parameter bandwidth, which has a physical meaning. Yet in the practical experiences, it is also hard to choose a proper one. And for the scikit-learn's implementation, the complexity tends towards $\mathcal{O}(T \cdot n \cdot log(n))$ in lower dimensions and $\mathcal{O}(T \cdot n^2)$ in higher dimensions, with n the number of samples and T the number of points [33].

```
Input: $DB$: Database; $\varepsilon$: Radius; $minPts$: Density threshold;
       $dist$: Distance function;
for each point $p$ in database $DB$ do
    if label(p) $\neq$ undefined then continue;
    Neighbours $N \leftarrow$ RangeQuery($DB, dist, p, \varepsilon$);
    if $|N| < minPts$ then
        label($p$) $\leftarrow$ Noise;
        continue;
    end
    $c \leftarrow$ next cluster label;
    label($p$) $\leftarrow c$;
    Seed set $S \leftarrow N \setminus \{p\}$;
    for each $q$ in $S$ do
        if label(q) = Noise then label($q$) $\leftarrow c$;
        if label(q) $\neq$ undefined then continue;
        Neighbors $N \leftarrow$ RangeQuery($DB, dist, q, \varepsilon$);
        label($q$) $\leftarrow c$;
        if $|N| < minPts$ then continue;
        $S \leftarrow S \cup N$;
    end
end

Function RangeQuery($DB, dist, q, \varepsilon$):
    Neighbors = empty list;
    for each point $p$ in database $DB$ do
        if $dist(q,p) \leq \varepsilon$ then
            Neighbors = Neighbors $\cup \{p\}$;
        end
    end
return Neighbors
```

Listing 2.1: Pseudocode of Original Sequential DBSCAN Algorithm. Adapted and summarized from [34, 35].

2.3.3 Density Based Clustering

Two density based clustering techniques are introduced and compared in this section: DBSCAN [34, 35] and its successor HDBSCAN [36, 37]. Again, there exists a number of variants with different implementation details like more efficient ones with different data structures to accelerate the performance, the original sequential version of DBSCAN [34] is introduced through pseudocodes as baseline. The pseudocodes of it can be found in Listing 2.1. In contrast, the [38]'s implementation for HDBSCAN is briefly introduced through the abstract pipeline together with several techniques of graph theory and data structures to emphasize the hierarchical characteristics.

The hierarchical variant of DBSCAN allows varying density clusters instead of predefined radius ε. As mentioned previously, the introduced DBSCAN algorithm is the original sequential version. In practice, it is realized with helps of several graph theory and data structure techniques. The pipeline below shows how HDBSCAN works, which is adapted from [38]'s implementation. Both two algorithms share the first three steps. Instead of taking radius ε as a threshold for the dendrogram as how DBSCAN works, however, a different approach is taken for HDBSCAN: the dendrogram is condensed by investigating the number of the points in the clusters. The pipeline of the HDBSCAN algorithm is introduced as below:

- Transform the space according to the density/sparsity. Based on the hyper-parameter `min_samples`, two terms are defined: *core distance* of the target point is the maximal distance between the target point and its `min_samples` nearest neighbors, and *mutual reachability distance* between two points is the maximal of three parts: their *core distances* and the distance between the two points, formalized as $d_{mr}(a, b) = \max\{d_{core}(a), d_{core}(b), d(a, b)\}$.

- Build the minimum spanning tree of the distance weighted graph. The calculated *mutual reachability distances* for point pairs can be considered as a weighted undirected graph. To build the minimum spanning tree, the *Prim's algorithm* is performed, which is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.

- Construct a cluster hierarchy of connected components. Given the minimum spanning tree, a single-linkage clustering is performed with the efficient union-find data structure to preliminarily hierarchize the points in the form of dendrogram. To this end, the difference between DBSCAN and HDBSCAN begins.

- Condense the cluster hierarchy based on minimum cluster size. Here, the second hyper-parameter `min_cluster_size` is in use. The cluster hierarchy is investigated in a top-down manner, and the splitting is taken place if both of the two split clusters have more than `min_cluster_size` points.

- Extract the stable clusters from the condensed tree. The strategy used to avoid artifacts is to omit all of its descendants if a cluster is selected. The measure of stability of a cluster is defined as $\sum_{p \in Cluster}(\lambda_p - \lambda_{birth})$, with $\lambda = 1/\text{distance}$. Intuitively speaking, the large distance of points inside one cluster implies that the cluster is unstable. With this in mind, if the stability of the target parent cluster is greater than the summed stabilities of all its descendants, the child clusters are unselected.

2.3.4 Mutex Watershed

Unlike the previously introduced clustering techniques, Mutex Watershed [39] is based on the graphs with weighted edges. As the name indicates, the proposed algorithm follows the analogue methodology of *Seeded Watershed* algorithm [40] and the clustering results are expected to be *mutual-exclusive (mutex)*, which means that each node belongs to one and only one cluster.

The algorithm is demonstrated with pseudocode as below:

```
Input: weighted graph $\mathcal{G}(V, E^+ \cup E^-, W^+ \cup W^-)$;
Output: clusters defined by active set $A^+$;
Initialisation: $A^+ = \emptyset$; $A^- = \emptyset$;
for $e \in (E^+ \cup E^-)$ in descending order of $W^+ \cup W^-$ do
    if $e \in E^+$ then
        if not connected$(i, j)$ and not mutex$(i, j)$ then
            merge$(i, j)$: $A^+ \leftarrow A^+ \cup e$;
        end
    else
        if not connected$(i, j)$ then
            addmutex$(i, j)$: $A^- \leftarrow A^- \cup e$;
        end
    end
end
```

Listing 2.2: Pseudocode of Mutex Watershed. Adapted from [39].
A self-implemented version using python library *NetworkX* [41] can be found at https://yuliwu.github.io/gist/mutex-watershed.

A weighted graph $\mathcal{G}$ is defined with the set of vertices V, the set of edges E and the set of weights W, where positive weights denote these two vertices are attractive to each other, and vice versa. All edges with two vertices `(i, j)` are traversed and added to the investigated set $A = A^+ \cup A^-$, where A^+ denotes the active set of positive edges and A^- denotes the mutual-exclusive set of negative edges. Two predication functions `connected(i,j)` and `mutex(i,j)` check whether two conditions are fulfilled:

- `connected(i,j)=True`: adding the edge `(i, j)` to the active set of positive edges A^+ causes a connected cycle in A^+;

- `mutex(i,j)=True`: adding the edge `(i, j)` to A^+ causes a connected cycle in $A^+ \cup A^-$, which contains at least one edge with a positive weight and at least one edge with a negative weight .

The corresponding operations are executed with different predications as presented in Listing 2.2:

- `merge(i,j)`: merge `(i, j)` and inherit the mutex constraints of the parent clusters;
- `addmutex(i,j)`: add mutex constraint between `(i, j)` .

2.4 State of the Art

In this section, two categories of instance segmentation algorithms are introduced. The first one utilizes the strategy called *instance-first*, where the bounding boxes are predicted in advance, and then the pixelwise instance-level labels (called *masks*) are refined based on the bounding boxes. Mask R-CNN [42] is a representative which follows the strategy. In Section 2.4.1, the modifications based on Mask R-CNN and other novel approaches with the same philosophy are also briefly introduced. The other category is so-called deep embedding learning algorithms, which follow *embedding and clustering* pipeline based on the embedding domain. Among those, four approaches [10–13] which share similar philosophies are investigated and compared in details.

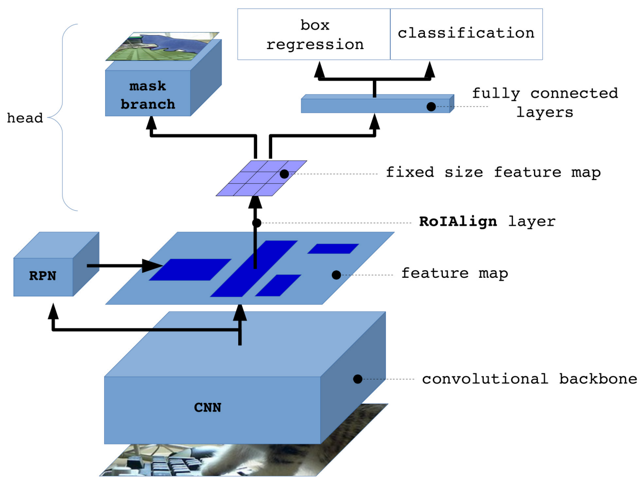
In addition, other approaches are worth briefly introduced. The following three methods are contour-based: [43] is focusing on learning the contours based on the semantic segmentations; [44,45] extend the *Watershed* and *Snake* algorithms, known as classical (non machine learning) image processing methods, to the deep learning versions respectively.

2.4.1 Instance-First Approaches

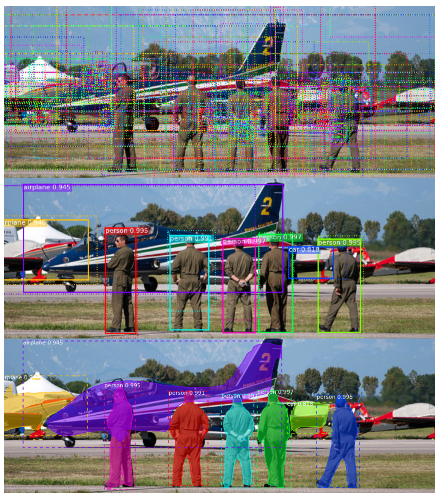
Instance-first means the bounding boxes of each instance-level objects are first detected, based on which the pixelwise masks are refined. In the following, two categories of methods are introduced, namely one with anchor-based object detector like *Faster R-CNN* [46] and the other with anchor-free object detector like *Fully Convolutional One-Stage* (FCOS) [47].

Faster R-CNN Based

The proposal-based meta-algorithm Mask R-CNN [42] has gathered a quite decent reputation in the field of instance segmentation due to its understandable end-to-end pipeline and reliable performance on the diverse datasets. The robustness of Mask R-CNN is partly attributed to the *Region Proposal Network* (RPN) originally from [46], which proposes instance-level candidates in the form of bounding boxes based on anchors. The pipeline is illustrated in Figure 2.8.



(a)



(b)

Figure 2.8: (a): Mask R-CNN model. Adapted from [48].
(b): Outputs from different stages. Top to bottom: proposal; box+class; mask. Adapted from [49].

To date, the Mask R-CNN and its variants has obtained a lot of success in the application field, especially those dealing with different challenges. Moreover, a considerable number of modifications based on Mask R-CNN have been proposed [50–52]. One of the most recent work is [53], which uses the techniques of classical computer graphics to improve the performance of mask refinement, by replacing mask head in the original work with the *Point-based Rendering* (PointRend) neural network module.

FCOS Based

Although the RPN provides robustness over varied image categories [49], the overhead of computation related to the anchor boxes can be avoidable if the bounding boxes can be detected directly from raw image inputs. FCOS [47] predicts per pixel *Classification, Center-ness and Regression* simultaneously in one stage, where *Classification* is denoted as a one-hot vector, *Center-ness* is denoted as the probability of this pixel being the center of a bounding box, and *Regression* is denoted as a one-by-four vector representing the distances between the center and the four edges of the bounding box. Due to its simpler pipeline compared to Faster R-CNN, FCOS is becoming the state-of-the-art object detector, and thus the first stage of instance segmentation to provide instance-level bounding boxes. Some of the instance segmentation approaches are directly concatenated after the FCOS pipeline, constructing two-stage approaches; while others modify the heads of FCOS, constructing single-shot approaches. The most recent former methods are shortly introduced below and the latter ones are introduced in Section 2.4.2.

[54] utilizes *Spatial Attention-Guided Mask* (SAG-Mask) to refine the masks based

on the predicted bounding boxes more accurately, as the *Spatial Attention Module* (SAM) helps to focus on the informative pixels and to reduce the noise. Furthermore, the light-weight backbone termed VoVNetV2, newly modified from VoVNet [55], makes this approach possible to perform in real-time. [56] exploits the *BlendMask* module, which shares the feature maps from backbone or FPN layers as bases, to segment position-sensitive instance features effectively. Only one convolution layer is added on top of the each bounding box prediction to produce attention maps, making this method real-time possible in inference.

2.4.2 One-Stage Approaches

As introduced previously, FCOS as a one-stage object detector simplifies the pipeline compared to Fast R-CNN and becomes popular as the first stage of instance segmentation. Moreover, FCOS can also be modified to become a one-stage instance segmentation method, without generating bounding boxes first.

In [57], the third head in original FCOS *Regression* is replaced by a one-by-n vector, where n denotes the number of rays which are the edges between the center and contour. Analogously, [58] is another single-shot instance segmentation method based on FCOS. In this work, the fourth head is added compared with original FCOS, termed *Mask Regression*, which is a compact representation of the encoded binary masks of the whole image by flattening, concatenation and reconstruction. In addition, the masks are calculated with help of pixel embedding and proposal embedding in [59], which are learned from the FCOS parallel to the original heads..

Table 2.1: Comparison of Deep Embedding Learning Methods.
s+learned denotes the embeddings are clustered using learned seeds and learned radii; s+hyper denotes the embeddings are clustered using learned seeds and self-defined angles as hyper-parameters.

Methods (by last name)	Distance	Local Constraint	Margin	Learned Seeds	Clustering
De Brabandere [11]	Euclidean		✓		Mean Shift
Payer [13]	Cosine	✓			HDBSCAN
Neven [12]	Euclidean		✓	✓	s+learned
Chen [10]	Cosine	✓		✓	s+hyper

2.4.3 Deep Embedding Learning Approaches

The approaches based on pixel embedding learning, which also belong to *one-stage* approaches, are becoming a new trend. In this section, four deep embedding learning approaches [10–13] are introduced. They share the general pipeline of *embedding* *and clustering.* Each pixel of input images is mapped to a high-dimensional vector (embedding), in which pixels of the same object are located closely. Then, clustering in the embedding space results in the final instance segmentation. De Brabandere and Neven [11, 12] have proposed Euclidean distance based embedding loss for instance segmentation. Payer et al. [13] have demonstrated embedding loss which utilizes cosine similarity and recurrent stacked hourglass network [19]. Chen et al. [10] have introduced a U-Net based architecture of two heads, where the embeddings are trained with cosine embedding loss and local constraints. These two heads are distance regression head and embedding head. The distance regression head aims to provide seed candidates for clustering. The proposed method inherits the fundamental modules from this work. A comparative table regarding with distinctive characteristics of the four methods is presented in Table 2.1.

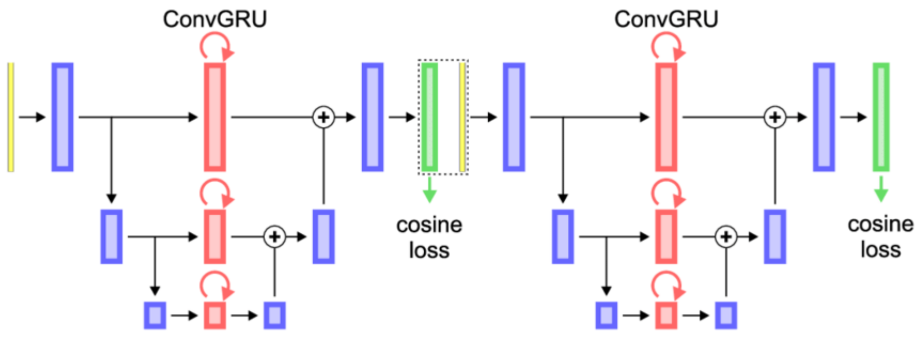


Figure 2.9: Hourglass with Cosine Embeddings. Adapted from [13].

2.4.4 Other Approaches on Leaf Segmentation

In this section, several approaches that focus on the dataset Leaf Segmentation are introduced of two groups.

The first one covers four classical (non deep learning) methods collectively reported in [4]. The IPK pipeline [60] relies on unsupervised clustering and distance maps to segment leaves. Another all-unsupervised method Nottingham achieves segmentation with SLIC superpixels [61], that does not require any training is used. The training dataset has been used for parameter tuning only. Moreover, the MSU approach [62] extends a multi-leaf alignment and tracking framework to the leaf segmentation based on Chamfer Matching [63]. Lastly, in Wageningen [4], the watershed algorithm [64] is applied on the foreground segmentation to achieve instance segmentation, which is obtained by supervised methods.

The second group is corresponding to the synthetic dataset, which can enlarge the number of training images and their shapes. Universal Plant Generator (UPGen) [65, 66] utilizes randomly sampled leaf geometries, textures and plant parameters to generate a 3D plant model for synthetic data. The augmentation of exploiting real+synthetic data brings in state-of-the-art results using Mask R-CNN.

2.5 Other Techniques

Distance Transform

Based on binary images as inputs, all pixels are transformed to their distances to the nearest zero pixels. Figure 2.10 illustrates an example.

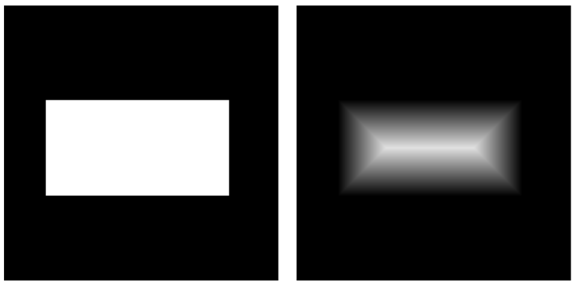


Figure 2.10: Example of Distance Transform.

In this thesis, the implementation of the OpenCV library with the algorithm from [67] is used. The transformed images are termed as *distmap* in the following chapters, from which the seeds (centroid pixels inside of one object) can be easily obtained after thresholding. Furthermore, the distmap can also be related to the trend of image segmentation term *attention*, which emphasizes merely the location of the target object and fades other pixels.

Sigmoid + Binary Cross-Entropy Loss

Originally designed for binary classification tasks, Binary Cross-Entropy Loss, or BCE Loss is defined as:

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{n=1}^{N} y_n^{gt}\log(\hat{y}_n) + (1 - y_n^{gt})\log(1 - \hat{y}_n), \tag{2.13}$$

where y_n^{gt} denotes ground truth and $\hat{y}_n$ denotes predictions.

For binary classification tasks, the outputs of the last layer before loss are usually activated with sigmoid function to make them bounded between 0 and 1, as shown in Figure 2.11(a). The sigmoid activation function is defined as:

$$\mathcal{F}_{Sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{2.14}$$

Linear/ReLU + Mean Squared Error Loss

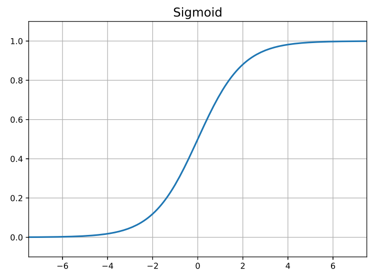
$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{n=1}^{N}(y_n^{gt} - \hat{y}_n)^2, \tag{2.15}$$

where y_n^{gt} denotes ground truth and $\hat{y}_n$ denotes predictions.

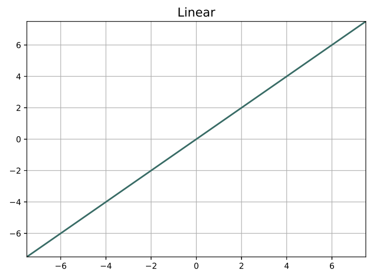
For regression tasks, the outputs of the last layer before loss are usually activated with linear function, as shown in Figure 2.11(b). In the case that the expected values are always non-negative, the *Rectified Linear Unit* (ReLU) can be also used to "manually" eliminate all negative signals, as shown in Figure 2.11(c). The linear and ReLU activation functions are defined as:

$$\mathcal{F}_{Linear}(x) = x \tag{2.16}$$

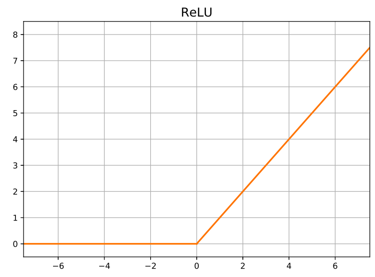
$$\mathcal{F}_{ReLU}(x) = [x]_+ = \begin{cases} 0, & \text{if } x < 0\,; \\ x, & \text{otherwise} \end{cases} . \tag{2.17}$$



(a)



(b)



(c)

Figure 2.11: Activation Functions: Sigmoid, Linear and ReLU.

mSBD: Mean Symmetric Best Dice

Dice score [68] is one of the most widely used metrics to evaluation quality of segmentation tasks. In the dataset CVPPP LSC, the Symmetric Best Dice (SBD) [4] is used, which illustrates the symmetric average Dice among all objects (leaves), where for each input label the ground truth label yielding maximum Dice is used for averaging, to estimate average leaf segmentation accuracy. Best Dice (BD) is defined as:

$$\mathrm{BD}(L^a, L^b) = \frac{1}{M} \sum_{i=1}^{M} \max_{1 \leq j \leq N} \frac{2|L_i^a \cap L_j^b|}{|L_i^a| + |L_j^b|}\,, \tag{2.18}$$

where $|\cdot|$ denotes leaf area (number of pixels) and L_i^a for $1 \leq i \leq M$ andL_j^b for$1 \leq j \leq N$ are set of leaf object segments belonging to leaf segmentations L^a and L^b, respectively. SBD between L^{gt}, the ground truth, and L^{ar}, the algorithmic result, is defined as:

$$\mathrm{SBD}(L^{ar}, L^{gt}) = \min\{\mathrm{BD}(L^{ar}, L^{gt}), \mathrm{BD}(L^{gt}, L^{ar})\}\,. \tag{2.19}$$

In this thesis, the Mean Symmetric Best Dice (mSBD) is used to evaluate the predicted results, which denotes the per image average of the SBD scores.

mAP: Mean Average Precision

Before introducing mean Average Precision (mAP) defined in [69], another evaluation metric analogue to Dice score, Intersection over Union is introduced with the definition:

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{FN} + \mathrm{TP} + \mathrm{FP}}\,, \tag{2.20}$$

where FN=False Negative, TP=True Positive and FP=False Positive. Note that the definitions here are corresponding to the number of pixels with perspective of objects.

Mean Average Precision (mAP) is calculated as follows. For each image, the predicted segmentation of each object is judged: if the IoU score for this predicted segmentation is larger than threshold IoU, then this predicted object-level segmentation is regarded as correct. After that, for each image the Precision can be calculated with respect to the number of FN, TP and FP predictions of objects:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{FN} + \mathrm{TP} + \mathrm{FP}}\,. \tag{2.21}$$

Note that the definitions here are corresponding to the number of predicted objects with perspective of images and it is different to widely accepted definition of *precision*, as FN is also taken into consideration. This processing is repeated for IoU=$\{0.5, 0.55, ..., 0.9\}$, and the calculated Precisions for each IoU threshold are averaged. As the name implies, *Average* denoted the averaged Precisions over IoU thresholds and *mean* denotes the averaged Average Precision over images.

3 Method

On the basis of the previously introduced state-of-the-art knowledge, a novel algorithm of instance segmentation via deep embedding learning, modified from [10], is presented in this chapter. It takes full advantage of learned distance transformation regression maps, which are originally used for extracting seeds for angular clustering during post-processing. In the experiments, it is found that the distance transformation regression maps being a concatenative layer to the original RGB images as the inputs of network can leverage the spatial information implicated in them to improve the accuracy of segmentation.

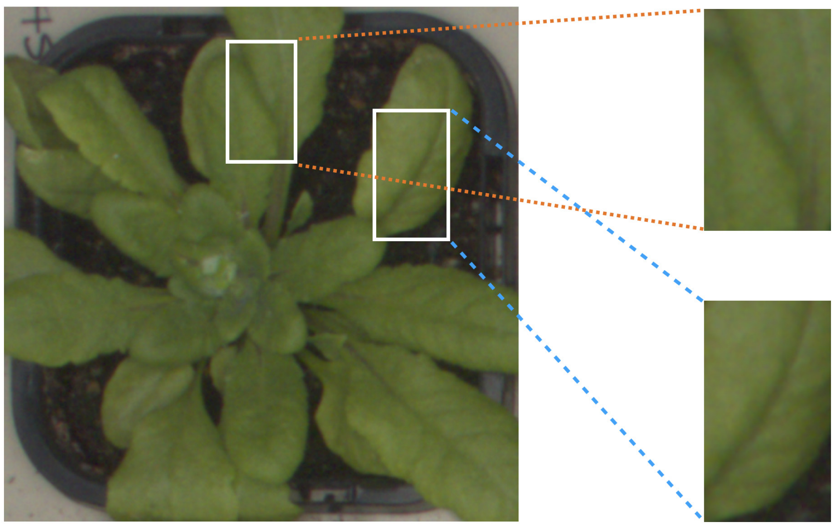


Figure 3.1: Ambiguity between Leaf Boundary and Leaf Midvein (Primary Vein).

One of the most challenging parts of the CVPPP Leaf Segmentation dataset is that the ambiguity between leaf boundaries and leaf midveins, as illustrated in Figure 3.1. The adjacent leaves can thus very hard to be correctly segmented, as the colors are extremely similar. This is also the problems reported from [10]. The main novelty of this thesis is corresponding to the module, so-called distance concatenative layer, which aims to supply direct and accurate spatial information. In this work, a simple, yet highly effective, architecture for object-aware embedding learning has been proposed. A distance regression module is incorporated into our architecture to generate seeds for fast clustering. At the same time, it is shown that the features learned by the distance regression module are able to promote the accuracy of learned object-aware embeddings significantly. By simply concatenating features of the distance regression module to the images as inputs of the embedding module, the

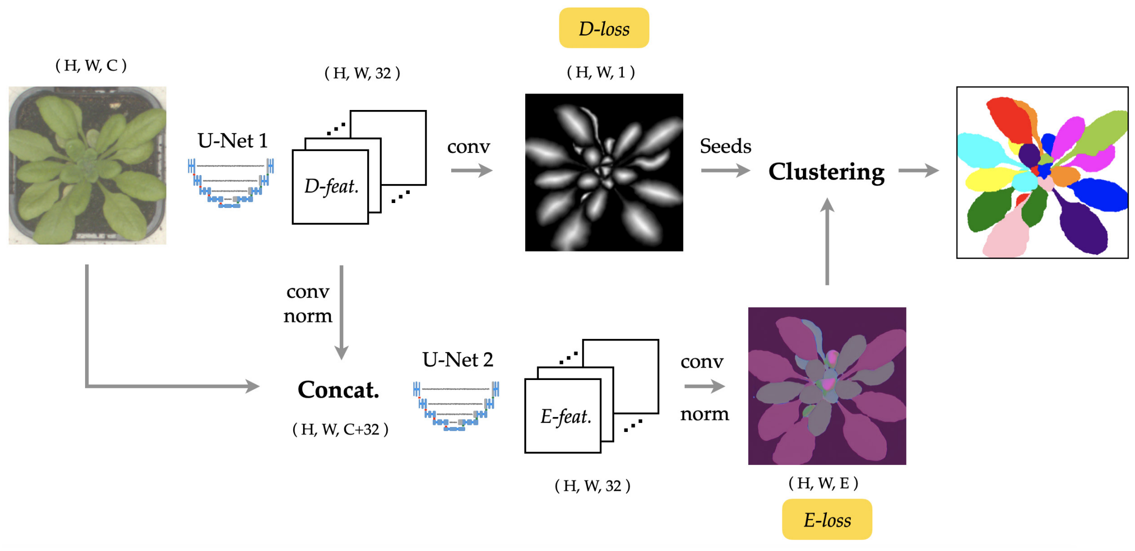


Figure 3.2: Processing Pipeline.
Distance regression features and distmaps are learned via distance module with U-Net 1. Concatenated distance regression features and raw images are fed into U-Net 2, from which the embeddings are learned. Final labels are generated based on seeds (thresholded maxima of distmaps) and embeddings via angular clustering. Denotations: H, W, C, E=Dimensions of Height, Width, Channel, Embedding.

mSBD scores on the CVPPP Leaf Segmentation Challenge can be further improved by more than 8% compared to the identical set-up without concatenation, yielding the best overall result amongst the leaderboard at CodaLab.

3.1 Processing Pipeline

The proposed network consists of two cascaded parts (Figure 3.2): the distance regression module and the embedding module. Each module uses a U-Net architecture with a 32-dimensional output feature map as the backbone network. The learned distance and embedding feature maps are denoted as *D-feat.* and *E-feat.*, respectively.

The distance regression module takes normalized images as the inputs and outputs the distance map (abbreviated as *distmap* in the following context) through a single convolutional layer with ReLU activation. The ground truth distmap is obtained by computing the shortest distances from pixels to the object boundary and then being normalized instance-wise with respect to the maximal value. The distance regression module is trained with Mean Squared Error (MSE) loss in this work, which is illustrated as *D-loss* in Figure 3.2.

Distance feature map *D-feat.* learned by the distance regression module is fed to the embedding module together with the input image by concatenation. Details of the concatenation are introduced in Section 3.3. The final embeddings are obtained through a convolutional layer with linear activation, followed by L2 normalization. The embedding module is trained with the loss based on the cosine similarity and local constraints (Section 3.2), denoted as *E-loss* in Figure 3.2.

The embedding space trained with loss in Equation 3.1 has a comprehensive geometric interpretation: embedding vectors of neighboring objects tend to be orthogonal, which simplifies the complexity of clustering. The fast *angular clustering* based on angles between embedding vectors can be effortlessly performed. Firstly, seeds are obtained from distmaps by fetching local maxima with a trivial threshold (selected as 70% of the global maximum in an image). After that, all neighboring pixels within the angular range δ_a of a seed are collected to form a cluster. In this work, $\delta_a = 45\,\text{deg}$ is used for all experiments. At last, the labels outside of the officially provided ground truth foreground masks are omitted.

3.2 Cosine Embedding Loss with Local Constraints

For the embedding module training, we build upon the loss format from [10]. The training loss, denoted as *E-loss* in Figure 3.2, is defined based on the cosine similarity $\mathcal{S}_{cos}(\mathbf{e_1}, \mathbf{e_2}) = \dfrac{\mathbf{e_1}^T\mathbf{e_2}}{\|\mathbf{e_1}\|\|\mathbf{e_2}\|}$ and is formularized as:

$$\begin{aligned}
\mathcal{L}_{emb} &= \lambda \cdot \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C}\sum_{c_A=1}^{C} \frac{1}{|\mathbf{N}_{c_A}|} \sum_{c_B \in \mathbf{N}_{c_A}} \Big[\mathcal{S}_{cos}(\boldsymbol{\mu_{c_A}}\,,\, \boldsymbol{\mu_{c_B}}) \Big]_+ \\
\mathcal{L}_{intra} &= \frac{1}{C}\sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[1 - \mathcal{S}_{cos}(\boldsymbol{e_i}\,,\, \boldsymbol{\mu_c}) \Big]_+ ,
\end{aligned} \tag{3.1}$$

where the embedding loss is defined as the weighted sum of the between-instance loss term $\mathcal{L}_{inter}$ and within-instance loss term $\mathcal{L}_{intra}$ with the weighting factor λ. $\boldsymbol{e}$ and $\boldsymbol{\mu}$ represents the pixel embedding vector and the mean embedding of an object, respectively. C denotes the number of objects, while the number of pixels of a single object c is denoted as E_c. $\mathbf{N}_{c_A}$ represents the set of neighboring objects around object c_A and $|\mathbf{N}_{c_A}|$ is the number of neighbors.

The between-instance loss term $\mathcal{L}_{inter}$ encourages the embeddings of different object to be separated, while the within-instance loss term $\mathcal{L}_{intra}$ punishes the case where pixel embeddings of the same object diverge from the mean. In addition, the local constraints of this loss only force neighboring objects to form separable clusters in the embedding space. The benefits of local constraints and the comparison with the global constraint are demonstrated in Section 4.2.2.

3.3 Feature Concatenative Layer

The feature map *D-feat.* learned by the distance regression module is firstly transformed to the desired dimension (shown with an example of 32 in Figure 3.2) via a convolutional layer and then L2 normalized along through each pixel before being concatenated to the raw images. Our experiment shows that the feature map normalization is critical to a stable training process.

As illustrated in Figure 3.1, the difference between leaf boundary and leaf midvein (primary vein) is ambiguous. The learned embeddings by the U-Net architecture [10] often fail at those locations. However, the distmaps are able to tell the difference with lower values representing leaf boundaries and higher values representing leaf midveins. From another perspective, the distmap, which gives an approximate outline of objects, can be interpreted as a *object-ness* score, the pixel-wise probability about existence of object. In addition, as proposed by [70], mixing convolutional operations with the pixel location helps constructing dense pixel embeddings that can separate object instances. From this perspective, the distance regression features can indirectly provide location information to the subsequent module. The performance improvement from U-Net with two heads to the proposed W-Net is illustrated. In Figure 3.3, two representative cases are demonstrated, where the U-Net fails to separate closely located leaves. In contrast, the W-Net has successfully distinguished the numbered leaves in Figure 3.3.

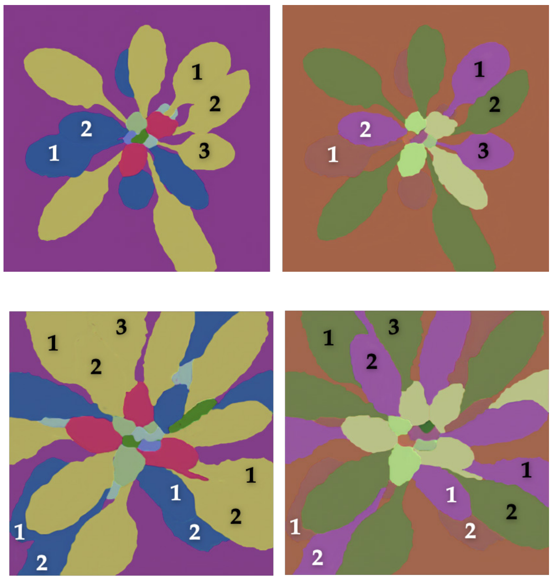


Figure 3.3: Learned Embeddings with U-Net and W-Net. Numbered leaves are treated as one object by U-Net, while they are successfully separated in the embedding space learned with W-Net.

To this end, a two-stage architecture has been conducted, as depicted in Figure 3.2, by forwarding the distance regression features to the embedding module.

And the concatenation of the distance regression features and raw images can bring in best performance in the experiments. We term the distance features as concatenative layer in between the stacked U-Nets as *intermediate distance regression supervision*. In the experiments, different features to forward have been tested: the 1-dimensional distmap, 8-dimensional distance features, 32-dimensional distance features, 32-dimensional embedding features, concatenated 16-dimensional distance features and 16-dimensional embedding features. Inspired by [12, 70], the performance of augmenting the input image with x- and y-coordinates has also been tested.

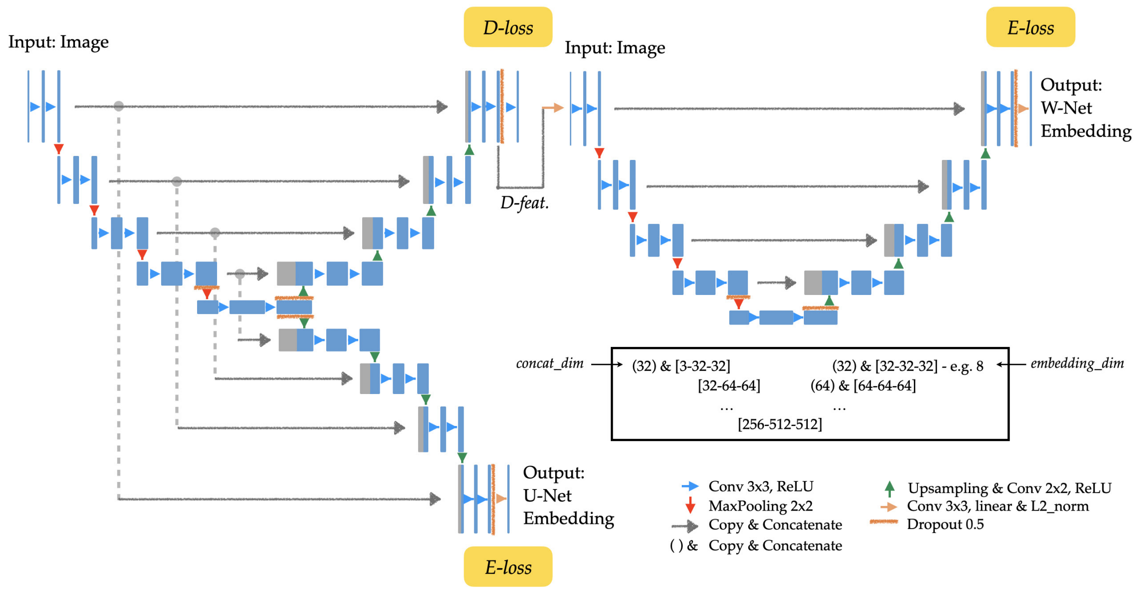


Figure 3.4: Hybrid Network Architectures of U-Net and W-Net.

3.4 From U-Net to W-Net

We abbreviate the proposed network as W-Net to differ from the existing U-Net with two heads, although the novelty and characteristic are not fully represented: the distance regression features as intermediate supervision and the cosine embedding loss with local constraints. In addition, a hybrid network (Figure 3.4) has been constructed, where the left half illustrates the U-Net with two heads from [10] and the top half illustrates the proposed one, termed W-Net. The intuition of this hybrid network is to fairly compare the network performance with and without the distance regression features in a simultaneous manner. By default, this hybrid network is used in experiments. Empirically speaking, the hybrid network results are consistent with that of the W-Net without the embedding head of the U-Net 1. The ablation experiments can be found in details in the next chapter.

4 Experiments

Based on the method presented in Chapter 3, the experiments related to the chosen set-ups are introduced in detail. Section 4.1 describes three datasets applied with the proposed method briefly. Following that, the main results are presented, including the top-positioned result of Leaf Segmentation Challenge. In Section 4.2, eight ablation experiments are conducted, which deliberates whether and to what extent the variables, including network architectures, loss formats and hyper-parameters, can exert an influence on the final results. Each section of ablation experiment consists of implementation and results subsections apart from the main introduction part, where the details of the parameter set-ups and corresponding comparison of final performance are showcased. The results subsections preliminarily discuss the influences of different components, while the overall and more detailed evaluation can be found in Chapter 5.

4.1 Datasets

In this thesis, three datasets are primarily used, which can be ostensibly categorized into two types: single-class and multi-class. CVPPP Leaf Segmentation and BBBC006: U2OS Cells are two datasets containing only one class [4, 71], and the Cityscapes dataset has multiple [9]. The focus of this thesis is to concentrate in the instance segmentation without considering the semantic segmentation. Therefore the single-class datasets are mainly investigated. With this in mind, the dataset of CVPPP Leaf Segmentation is the target dataset in this thesis, and the dataset of BBBC006: U2OS Cells is served as validation also with final scores to evalu-

Table 4.1: Roles of 3 Datasets in this Thesis.
Class: the kind of objects; Channel: RGB or grayscale; Ablation: the comparison experiments, see Section 4.2; Scores: concrete evaluation scores, like Dice or AP; Predict: showcases of predicted instance segmentation labels; Testing gt: whether the ground truth of testing set is given.

	Class	Channel	Ablation	Scores	Predict	Testing gt
Leaf Seg.	single	3	✓	✓	✓	unknown
U2OS Cells	single	1		✓	✓	known
Cityscapes	multi	3			✓	unknown

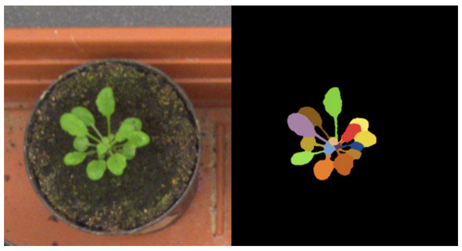
ate the performance. The dataset of Cityscapes, due to its complexity of semantic segmentation, is merely demonstrated by the predicted labels as showcases without comparing the concrete scores for evaluation. Table 4.1 showcases the main features and roles of the three datasets in this thesis.

4.1.1 CVPPP Leaf Segmentation

Computer Vision Problems in Plant Phenotyping (CVPPP) holds several biological and agricultural datasets, including Leaf Segmentation [4], Leaf Counting [4] and Global Wheat Head Detection [72]. Leaf Segmentation and Leaf Counting are hold as workshops or challenges by top-level conferences in the recent years, including ECCV 2014 (Zurich, Switzerland), BMVC 2015 (Swansea, UK), ICCV 2017 (Venice, Italy), BMVC 2018 (Newcastle, UK), CVPR 2019 (Long Beach, USA) and upcoming ECCV 2020 (Glasgow, UK).

Since 2017, the Leaf Segmentation dataset has extended with more images and a permanent challenge website has been set up in CodaLab: https://competitions.codalab.org/competitions/18405. This is the version used in this thesis due to its fairness, as the ground truths of testing set are not available and the scores can be learned only by submitting the results.

The Leaf Segmentation dataset contains two types of plants: Arabidopsis and Tobacco, as shown in Figure 4.1. They are particularly different speaking of shapes, including the size of leaves and the leaf stems. As captioned, one of the challenging tasks of this dataset is the extremely imbalanced number of the two types of plants in training set and testing set. Specifically, the images of Tobacco contribute only approximately 3% in training set, whereas more than 20% of testing images are Tobacco.


(a) Arabidopsis: 783 train / 388 test

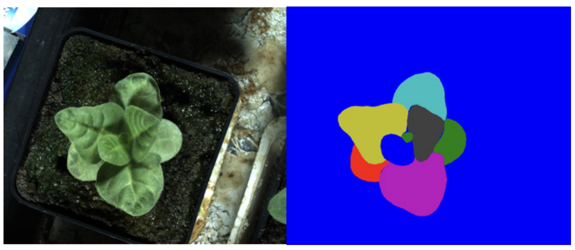
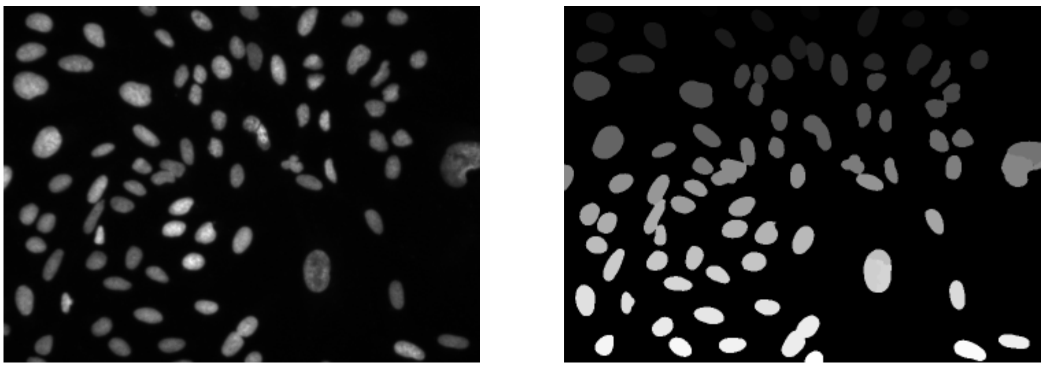

(b) Tobacco: 27 train / 112 test

Figure 4.1: Examples of Leaf Segmentation.

4.1.2 BBBC006: U2OS Cells

Broad Bioimage Benchmark Collection (BBBC) contains a rich range of microscopy image sets provided by Broad Institute [71]. The used dataset BBBC006: U2OS Cells is one of them that consists of Human Bone Osteosarcoma Epithelial Cells, also known as U2OS Cells. Totally 754 images are taken for the instance segmentation, half among which are categorized into the training and validation set with the ratio 4:1 and the other half into the testing set. The grayscale images are transformed to the shapes (512,512) in the datatype of `uint16`. Unlike the Leaf Segmentation where the testing ground truth is unreachable and the evaluation for predicted results is completed totally online, the evaluation for this dataset is locally carried out. The purpose of using this dataset is to compare different set-ups based on the prior knowledge of ablation experiments using Leaf Segmentation. The results can be found in Section 4.4.



(a) **(b)**

Figure 4.2: An Example of BBBC006: U2OS Cells

4.1.3 Cityscapes

The *Instance-Level Semantic Labeling Task* is one of the four benchmarks of Cityscapes dataset. The following ten semantic classes have instance-level labels available: person, rider, car, truck, bus, train, motorcycle, bicycle, and caravan, trailer. The predicted instance-level segmentations of last two classes, caravan and trailer, are not considered in the final submission. Also to notice that if the boundary between instances cannot be clearly seen, the whole crowd is labeled together and annotated as group, e.g. car group.

As shown in Figure 4.3, the Cityscapes dataset is more complicated than previous two. The cars and persons are often occluded, and the sizes of the objects are rather varied. Furthermore, due to the point of view from the cockpit, the depth of field and the vanishing point in the middle are salient. The size of images plays also a significant role in the training, as too large input images may be out of memory. The image shape of (1024, 2048) is much larger than that of Leaf dataset with the average shape of (512, 512). This means the original images should be cropped into patches and be trained patch-wise, as simply rescaling may lose the accuracy and details, especially some of far cars and persons are extremely tiny.

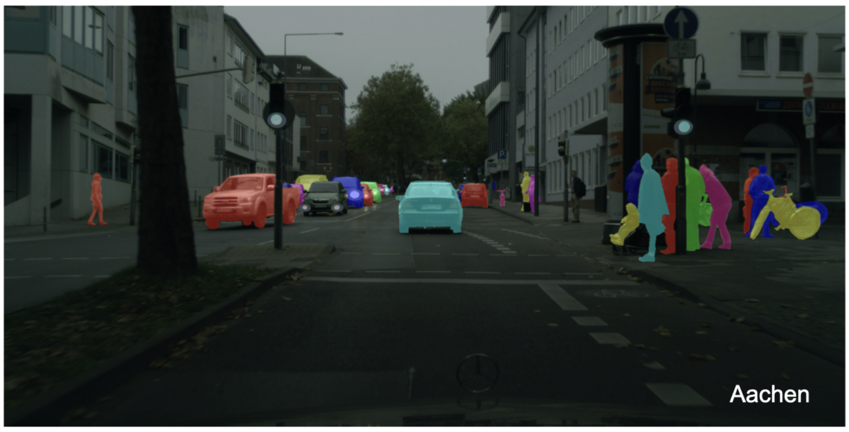


Figure 4.3: An Example of Cityscapes.

4.2 Ablation Experiments

In this section, eight ablation experiments are conducted to investigate the potential impact factors on the final performance. Before the detailed experiments and their evaluations, the general set-ups are introduced. The default network is W-Net with Intermediate Supervision with local constraints in the format of polar embedding loss. The loss of distance regression is ReLU+MSE and the concatenative layer is the output of last feature map layer from the first U-Net. If the set-ups are changed in the experiments specifically, details are presented in the Implementation subsections.

Following are the details of training technique set-ups for all experiments: The weights initializer is He Normal [73] and the optimizer is Adam [74]. The adaptive learning rate is used with exponential decay, where the initialized learning rate is set to 0.0001, the decay steps are set to 5000 steps and the decay rate is set to 0.9. Without specifically mentioning, batch size is set to 4. In some experiments, the batch size is set to 2 to avoid being out of memory because larger images are used. The ablation experiments cannot traverse all possible parameters, only some representatives are chosen to demonstrate the different performance. The set-ups of different ablation experiments can be also varied, yet the general expectation can be realized to utilize optimal parameters from each single ablation experiment to obtain possible best performance.

4.2.1 Cartesian vs. Polar Embedding

As introduced and compared in Section 2.2.2 - Section 2.2.3, two main loss formats can be applied in deep embedding learning approaches. The Cartesian Form benefits from its intuitive implementation, the embeddings of which are however not ranged. The unlimited embedding domain in the case of Cartesian Form has also advantages: it is able to work in 2d, only the most remote embeddings may be located far away from the origin. The euclidean embeddings in the Cartesian Form can also be scaled to manually set ranged, which brings two drawbacks:

- The high precision of embeddings is demanded;
- The benefit of local constraints cannot be effectively exploited.

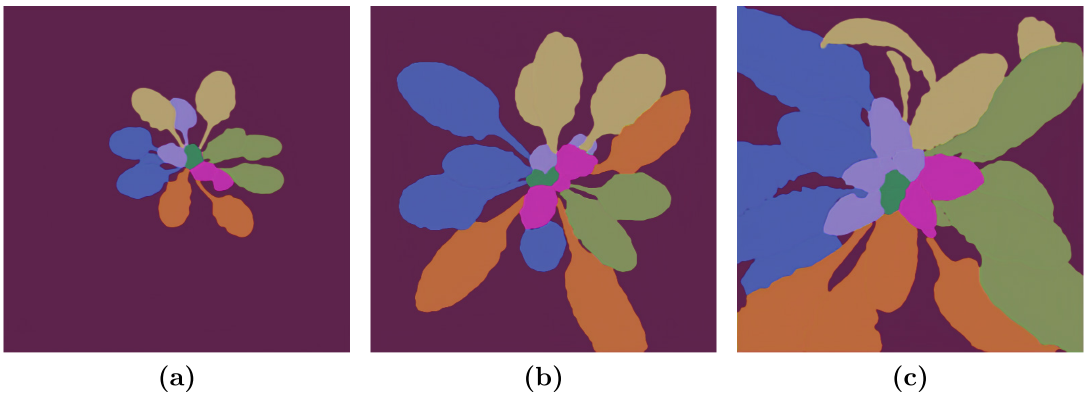


Figure 4.4: Examples of Euclidean Embeddings with Global Constraints. Images are training set results with 8 dim embeddings. Totally 8 different colors including background have been generated.

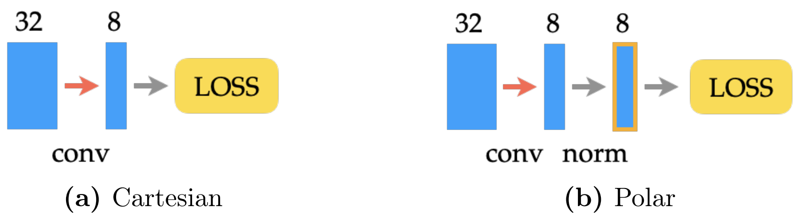
Compared to the Cartesian Form, the Polar Form has the advantage of *circulation*. Take an $n-$dim unit ball as an example, totally n linearly independent basis vectors can be found. If the optimized progress is to pick one vector, such that this vector is orthogonal to existed several vectors, a *circulation* occurs. The n basis vectors are picked one by one, as they are all orthogonal to each other. After all basis vectors have been once selected, the previously selected basis vectors may be picked again, as they are far away from the last pick. The *circulation* of Polar Form enables the representation of instance objects with rather limited number of basis vectors. In the instance segmentation tasks, the distribution of embeddings, or more informally speaking coloring the instance objects, can be vividly described by the discussed progress of picking basis vectors. This phenomenon is illustrated in Figure 4.4, where three resulting images of different sizes from training set using global constraints are shown. The embeddings are of 8 dims, the first 3 of them are illustrated. Very interestingly, there are totally 8 colors, which corroborates the previous assumption corresponding to the basis vectors.

To clarify how the different components in the network architectures are termed in this thesis:

- The backbone has two versions: U-Net and U-Net with 2 heads, see Appendix A for details;
- One head is defined as the right bottom-up half pathway of original U-Net;
- The set-ups of backbones are identical in all experiments with the first and the last embeddings (or feature maps) of 32 dims;

- The final embeddings are transformed to the selected dimensions via a tiny convolutional layer block, termed joint block between backbone and loss in this thesis to avoid confusion with heads.

In Figure 4.5, the different settings of joint blocks between backbone and loss for Cartesian Form and Polar Form are illustrated. The final embeddings are defined as 8 dim in this example. The `dropout` layer with `rate=0.5` is applied in the first convolutional layer. One significant difference between two forms is the application of L2 normalization layer. No doubt that the euclidean embeddings should not be normalized, as the Euclidean distance would be destroyed after normalization.



(a) Cartesian **(b)** Polar

Figure 4.5: Joint Block between Backbone and Loss.
Numbers above denote dimensions. Red arrows denote `dropout`.

As previously mentioned, the first 3 dims of total 8 dim embeddings are shown as RGB channels. Empirically, it is capable to observe the performance of the embeddings. In this thesis, a number of images using this abbreviated representation of embeddings can be found. As the embedding learning progress starts with randomly initiated weights and the training images are out of order, for each trained model there is a different color map for illustrating embeddings: also enjoy the artistic color palettes made by machine!

The detailed loss functions are formalized in Table 4.2. The implementation and results subsections are jointly presented in the following ablation experiment about local and global constraints in Section 4.2.2.

4.2.2 Local vs. Global Constraints

The most salient difference between semantic segmentation and instance segmentation (specifically speaking one-class instance segmentation) is that the number of semantic labels are usually given in advance, while the number of instances not. Therefore, it is much harder to give each object a distinguished label. In addition to this methodology, the progress of label distribution can be carried out locally: different labels are hanged out only to adjacent objects. The former methodology is termed global constraints and the latter one is termed local constraints in this thesis. The mathematical background behind this idea is the Four Color Theorem [25].

Training using local constraints is not only advantageous. It makes the training more difficult, as naturally speaking more detailed spatial information is required.

After the training progress, the network should also tell if the objects are adjacent to each other, which reflects the title of this thesis where the *densely* located objects are emphasized.

Table 4.2: Four Formats of Loss Functions.
The following four equations depict the four cases of this experiment: (Cartesian Loss, Polar Loss) × (Global Constraints, Local Constraints).

	Cartesian	Polar
Global	Eq. 2.4	Eq. 4.2
Local	Eq. 4.1	Eq. 2.5

$$\begin{aligned}
\mathcal{L}_{emb} &= \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C(C-1)} \sum_{c_A=1}^{C} \sum_{\substack{c_B=1 \\ c_A \neq c_B}}^{C} \Big[D_{euc}(\boldsymbol{\mu_{c_A}} \,,\, \boldsymbol{\mu_{c_B}}) - 2\delta_1 \Big]_+^2 \\
\mathcal{L}_{intra} &= \frac{1}{C} \sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[\delta_2 - D_{euc}(\boldsymbol{e_i} \,,\, \boldsymbol{\mu_c}) \Big]_+^2
\end{aligned} \tag{2.4 revisited}$$

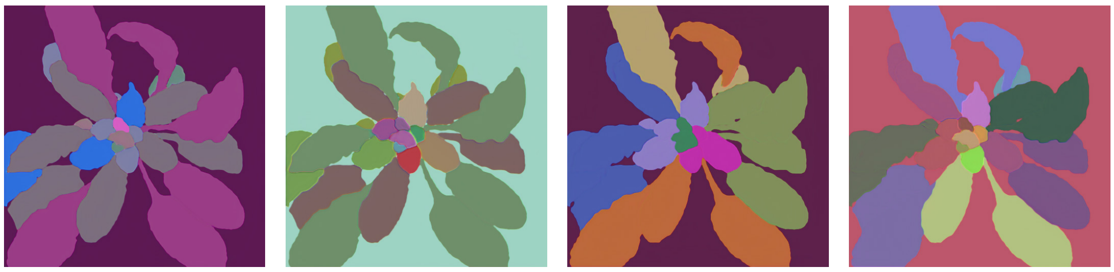
$$\begin{aligned}
\mathcal{L}_{emb} &= \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C} \sum_{c_A=1}^{C} \frac{1}{|\mathbf{N}_{c_A}|} \sum_{c_B \in \mathbf{N}_{c_A}} \Big[1 - D_{cos}(\boldsymbol{\mu_{c_A}} \,,\, \boldsymbol{\mu_{c_B}}) \Big] \\
\mathcal{L}_{intra} &= \frac{1}{C} \sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[D_{cos}(\boldsymbol{e_i} \,,\, \boldsymbol{\mu_c}) \Big]
\end{aligned} \tag{2.5 revisited}$$

$$\begin{aligned}
\mathcal{L}_{emb} &= \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C} \sum_{c_A=1}^{C} \frac{1}{|\mathbf{N}_{c_A}|} \sum_{c_B \in \mathbf{N}_{c_A}} \Big[D_{euc}(\boldsymbol{\mu_{c_A}} \,,\, \boldsymbol{\mu_{c_B}}) - 2\delta_1 \Big]_+^2 \\
\mathcal{L}_{intra} &= \frac{1}{C} \sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[\delta_2 - D_{euc}(\boldsymbol{e_i} \,,\, \boldsymbol{\mu_c}) \Big]_+^2
\end{aligned} \tag{4.1}$$

$$\begin{aligned}
\mathcal{L}_{emb} &= \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C(C-1)} \sum_{c_A=1}^{C} \sum_{\substack{c_B=1 \\ c_A \neq c_B}}^{C} \Big[1 - D_{cos}(\boldsymbol{\mu_{c_A}} \,,\, \boldsymbol{\mu_{c_B}}) \Big] \\
\mathcal{L}_{intra} &= \frac{1}{C} \sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[D_{cos}(\boldsymbol{e_i} \,,\, \boldsymbol{\mu_c}) \Big]
\end{aligned} \tag{4.2}$$

Implementation

The neighborhood is defined in this experiment as follows: all other objects which have at least one pixel not farther than a predefined distance as hyper-parameter to at least one pixel of the target object are considered as its neighbors. The distance is selected as 2% of the shorter edge from height and width of the input image.



(a) local 8 **(b)** local 64 **(c)** global 8 **(d)** global 64

Figure 4.6: Learned Embeddings for Combined Cases of Local/Global Constraints and 8/64-dimensional Embeddings. (a,b) vs. (c,d): Local constraints ensure the effective utilization of embedding domain, as same embeddings appear alternately for non-adjacent objects. (a) vs. (b): Higher-dimensional embeddings are redundant in the local constraint case. (c) vs. (d): Lower-dimensional embeddings with global constraints are not sufficient to distinguish all objects. This problem is slightly mitigated via higher-dimensional embeddings, still not as effective as local constraints.

Results

Local constraints make it possible to exploit lower-dimensional embedding space more efficiently, as in this case, we only distribute different labels to neighboring objects. In contrast, the global constraints have to thoroughly give each single object in the images a different label, which requires larger receptive fields and more redundant embedding space. The combination of local constraints and cosine embeddings utilizes the embedding space further comprehensively, as the push force imposed by loss expects orthogonal embedding clusters for neighboring instances.

This is confirmed qualitatively by examples showcased in Figure 4.6. In Figure 4.6(c), 8-dimensional embeddings are trained with global constraints. Not surprisingly, there are exactly 8 colors in the image, indicating 8 orthogonal clusters in the embedding space. Apparently, the global constraint will fail when the embedding dimension is fewer than the number of objects. In contrast, the local constraints (Figure 4.6(a) - 4.6(b)) can distribute labels alternately between objects, with the same labels appearing multiple times for non-adjacent objects. This makes it possible to utilize a lower-dimensional embedding space. Quantitatively, the W-Net trained with local constraints surpasses the one trained with global constraints by more than 4% on overall mSBD, as listed in Table 4.6.

Intuitively, a higher-dimensional embedding space is able to provide a higher degree of freedom, i.e. we can simply use higher-dimensional embeddings to alleviate the problem of global constraints. At least the embedding vector does not have to be restricted to low dimensions. However, from the results in Figure 4.16, we find that higher-dimensional embeddings produce worse results. This makes the capability of using lower-dimensional embedding space particularly important.

4.2.3 U-Net vs. W-Net

Inspired by [13,19], the stacked variant of U-Net can be advantageous to improve the performance. To describe the networks vividly, the two variants are termed U-Net and W-Net, where the latter network is designed as two stacked U-Net. Figure 4.7 illustrates the simplified architectures, and the more detailed version can be found in Appendix A. Without loss of generality only up to two U-Nets are stacked. Since the distance regression has to be added to the network to calculate the seeds for angular clustering in the post-processing, three variants are compared, as Figure 4.7(b) - Figure 4.7(d) show. In the following, apart from the original U-Net, three variants of it are introduced: U-net with 2 Heads, W-Net without Intermediate Supervision and W-Net with Intermediate Supervision.

U-net with 2 Heads

Originated from [10], this network has 2 separate and parallel upsampling paths, termed *heads* in this thesis, for both distance regression and embedding loss. It utilizes the downsampling path efficiently by sharing the features to the two heads. The two loss functions are then summed. The details of the network can be found in Figure 4.7(b) under Appendix A. There are two reasons for this kind of architecture:

- The distance regression head is necessary, as the seeds for angular clustering in the post-processing should be calculated by the learned distmap;
- The shared first half top-down pathway of the U-Net can reduce computational overheads. The parallel layout of the distance regression and embedding loss head can exploit the advantage of parallel computation.

After each head, the joint block between backbone and loss is stacked, see the examples shown in Figure 4.5. The joint block for distance regression is analogous to the Cartesian one, only with 1 dim convolutional layer as the distmap is mono-channel.

W-Net: 2 Stacked U-Nets

The stacked version of U-Net provides the ability of concatenative layer between two U-Nets. Why the concatenative layer is useful to extend features can be deduced by analogy of RGB images against grayscale ones, as the concatenative construction is identical. The choice of concatenation over addition is also comprehensible by the example of RGB and grayscale images, as the concatenated tensors can be simply

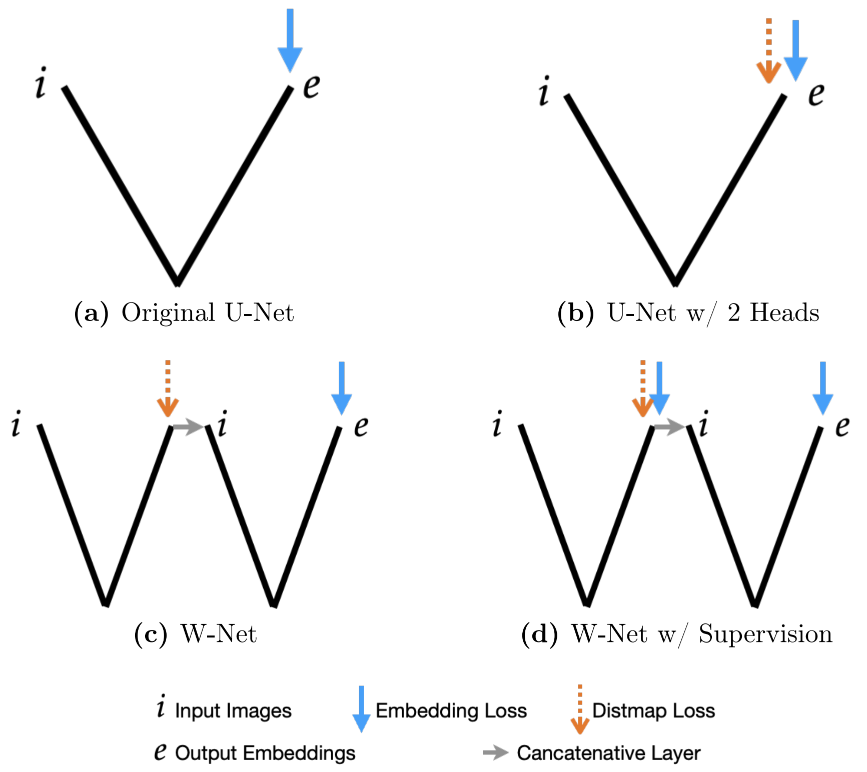


Figure 4.7: Simplified Illustrations of U-Net and Variants.
The complete network architectures can be found in Appendix A.

operated by convolutional layer to the additive results, while the addition cannot be transformed to concatenative results effortlessly.

If the stacked U-Nets are used, one nature modification is to put distance regression head at the end of the first U-Net. The reasons are as follows:

- Separately located two heads are advantageous for concentrated training of both the distance regression branch and embedding branch, as they have full-sized U-Nets respectively;
- The feature maps of distance regression head are intuitively full of spatial information, as distmap contains suggestions of where the instances are located and the *object-ness*, the probability of existence of objects.

W-Net with Intermediate Supervision

[19] shows that the performance can be further improved by adding *Intermediate Supervision*, which is regarded to as the additive loss in the early stages of stacked U-Nets. *Intermediate Supervision* helps to keep the longer learning progress in control. With this in mind, the first U-Net from the previously presented W-Net is modified to the U-Net with 2 Heads, where the embedding loss is also calculated intermediately. Again, the concatenative layer stays identical, namely the feature map

of distance regression. The detailed investigation of the influences of concatenative layer can be found in Section 4.2.4.

Table 4.3: Comparison of U-Net Variants.
Abbreviations: D=Distance regression supervision; D+E=Distance and Embedding supervision.

Architectures	Number of U-Net	Parallel Heads	Intermediate Supervision	Concatenative Layer
U-Net	1			
U-Net w/ 2 Heads	1	✓		
W-Net	2	✓	✓	D
W-Net w/ Supervision	2	✓	✓	D+E

Implementation

A summary of the network architectures can be found in Table 4.3. This experiment is based on the last three variants, as the distance regression is required for the angular clustering. All trainings use the Polar loss format with local constraints for embedding loss and ReLU+MSE for distance regression loss. If applicable, the concatenative layer is the 32 dim feature map of distance regression. All embeddings are transformed to 8 dims before calculating losses. The two loss terms are equally contributed with constant weights 1. All losses are summed, including the intermediate embedding loss (only applicable for W-Net with Supervision), distance regression loss and final embedding loss.

Results

Firstly, the performance improvement from U-Net with two heads to the proposed W-Net has been illustrated via the learned embeddings in Figure 3.3, where two representative cases are demonstrated, in which circumstances the U-Net fails to separate closely located leaves. In contrast, the W-Net has successfully distinguished the numbered leaves in Figure 3.3. This can be confirmed by the results of segmentations on the testing set. Figure 4.8 illustrates some salient examples, where the W-Net has segmented more accurately than U-Net.

Quantitatively, W-Net surpasses U-Net on overall mSBD by approximately 8% from 0.794 to 0.879 with the best set-ups for W-Net, as shown in Table 4.4. Under different settings of embedding dimensions (Figure 4.16) and loss weights (Figure 4.10), the performance gap between U-Net and W-Net can be continuously observed and remain about 8%.

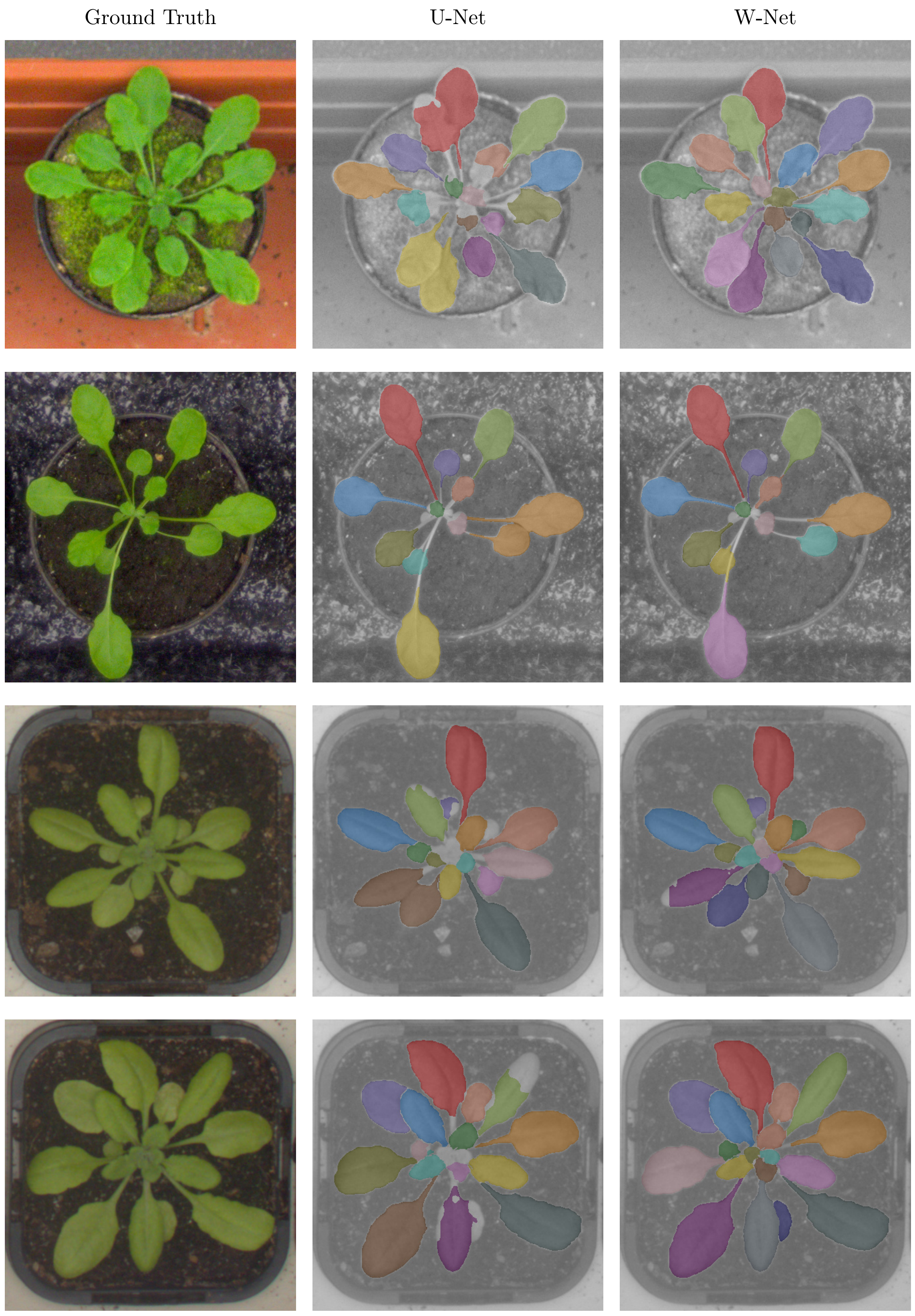


Figure 4.8: Leaf Segmentation Results of the CVPPP2017 Testing Set: Ground Truth (left), U-Net (middle) and W-Net (right); Each row illustrates one example. Improvement from U-Net to W-Net is salient. The mSBD score has increased from 0.794 to 0.879.

4.2.4 Concatenative Layer

Giving the machine a target feature relevant hint is intuitively beneficial to the faster convergence. In [10], the U-Net with 2 Heads (Figure 4.7(b)) is used as backbone. To explain the preference of concatenation over summation, the example of RGB images and grayscale images can be analogously considered. As the concatenated tensors can be simply operated by convolutional layer to the additive results, while the addition cannot be transformed to concatenative results effortlessly. In other words, the concatenative layer can be regarded as a generalized version of summation with learned weights instead of constant 1. Some flawed results are related to the problem that the adjacent leaves cannot be segmented separately. The other salient problem is that the extremely large and densely located leaves cannot be segmented correctly. Since one of the to be solved problems is the occasionally failed segmentation of adjacent leaves, the concatenative layer which can provide rich spatial information is expected.

As the distmaps are trained together with embeddings, to calculate seeds for angular clustering in the post-processing, which makes the distmap a natural choice for the concatenative layer. Distmap offers following information:

- Centers of instances can be calculated using distmap;
- Probability of if here exists an instance. Or *object-ness* as in some papers.
- Distinguish the leaf boundary and leaf midvein by marking the boundary as low values and marking the midvein as high. An example of this problem can be found in Figure 3.1. The effect is demonstrated in Figure 4.9.

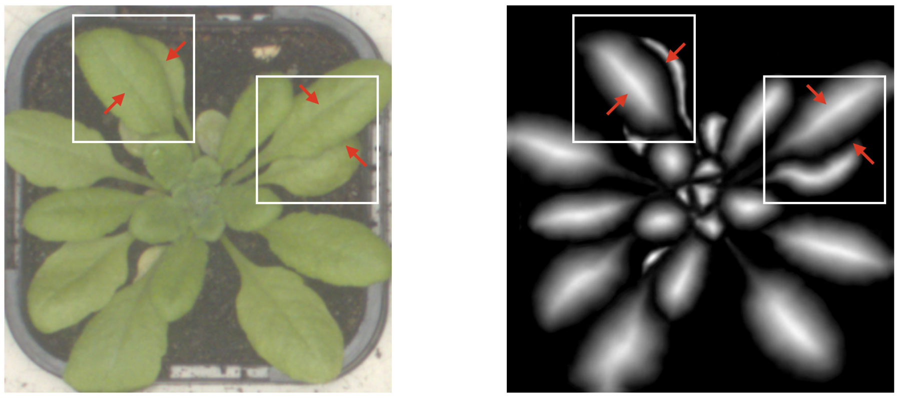


Figure 4.9: Distmap as Concatenative Layer.
Red arrowed areas indicate the leaf boundary and leaf midvein, which can be distinguished by distmap. The boundary is marked as low values and the midvein as high.

The other candidates of concatenative layer are as follows:

- The feature maps of distmaps with higher dimensions. The more abstract features are worth being experimented.
- The coordinates, which consist of two dimensions representing X-axis and Y-axis. As the desired hints are related to the spatial information, this is a natural proposal.
- The feature maps of embeddings from the first U-Net. The performance of the second U-Net may be improved by synthesizing the trained embeddings and raw images.
- The stacked feature maps of distmaps and of embeddings. It remains to be verified, if richer information leads to better performance.

Implementation

We compare the effects of different types of concatenative layer. Firstly, the distmap (1-dimensional) can be directly forwarded. Alternatively, the distance regression features instead of the distmap can be utilized. Before concatenation, we convert the 32-channel *D-feat.* into 8 and 32 dimensions (denoted as *dfeat.8* and *dfeat.32* in Table 4.4) through a single convolutional layer. Meanwhile, we have also tested the case of using embedding loss as the intermediate supervision (*efeat.32*). Specifically, the embedding features from the first U-Net are concatenated with the images as embedding module inputs. Furthermore, the distance regression features and embedding features (*dfeat.16+efeat.16*) are also investigated. At last, augmenting the input image with coordinates is tested. As proposed in [70], constructing dense object-aware pixel embeddings cannot be easily achieved using convolutions and the situation can be improved by incorporating information about the pixel location. In this work, we augment the input image with two coordinate channels for the normalized x- and y-coordinates, respectively. A similar implementation has also been used in [12].

Results

Experimental results are summarized in Table 4.4. First of all, forwarding distmaps is not as effective as forwarding feature maps, including the distance regression features and the embedding features. The embedding features (*efeat.32*) can also boost the performance, but not as significantly as the distance regression features. This is verified by the fact that *efeat.32* is worse than *dfeat.32* and the mixed feature map *dfeat.16+efeat.16*. For the distance regression feature itself, higher dimension of 32 is preferred. Finally, augmenting images with coordinates does not show apparent differences in our experiments. The effects could be further studied. For example, augmenting each intermediate feature map with coordinates is also worth being investigated.

Table 4.4: Comparison of Different Types of Concatenative Layers. Denotation: dfeat.16+efeat.16 = concatenated distance features of 16 dim and embedding features of 16 dim. Others can be analogously educed.

Concatenative Layer	Net	mSBD
none (baseline)	U-Net	.794
coordinate	U-Net	.798
distmap	W-Net	.824
dfeat.8	W-Net	.864
dfeat.32	W-Net	**.879**
efeat.32	W-Net	.847
dfeat.16+efeat.16	W-Net	.873

4.2.5 Loss Weights as Hyper-Parameters

The weighting factors of both $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ are previously set to 1. It is worth investigating whether the weights have an influence on the optimization progress and final results, as higher weight indicates that the term has more dynamics during the optimization. With this in mind, the only change in this experiment is the weighting factor of between-instance loss term. It is formularized by replacing the addition of equally contributed $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ in Equation 2.5:

$$\begin{aligned}
\mathcal{L}_{emb} &= \lambda \cdot \mathcal{L}_{inter} + \mathcal{L}_{intra} \\
\mathcal{L}_{inter} &= \frac{1}{C}\sum_{c_A=1}^{C} \frac{1}{|\mathbf{N}_{c_A}|} \sum_{c_B \in \mathbf{N}_{c_A}} \Big[1 - D_{cos}(\boldsymbol{\mu_{c_A}} , \boldsymbol{\mu_{c_B}}) \Big] \\
\mathcal{L}_{intra} &= \frac{1}{C}\sum_{c=1}^{C} \frac{1}{E_c} \sum_{i=1}^{E_c} \Big[D_{cos}(\boldsymbol{e_i} , \boldsymbol{\mu_c}) \Big]
\end{aligned} \tag{4.3}$$

Since one of the to be solved problems is the occasionally failed segmentation of adjacent leaves, the hypothetical cause is the within-instance loss $\mathcal{L}_{intra}$ has too many influences on the embeddings. Based on this assumption, the tendency of setting the between-instance loss weight λ is larger than 1, to amplify the effect of $\mathcal{L}_{inter}$. After revisiting the two loss terms in Equation 4.3, one salient difference between the two terms is that all impacts upon the individual embeddings in $\mathcal{L}_{inter}$ are obtained via their cluster centers $\boldsymbol{\mu_c}$ after being applied average operation, while the direct control on the individual embeddings $\boldsymbol{e_i}$ exists in the within-instance loss term $\mathcal{L}_{intra}$. With this in mind, the dynamics of the embeddings during the optimization progress from gradient descent of two losses are *imbalanced*.

This problem is corresponding to the fundamental question of whether this loss format is effective for the embedding learning. Therefore, two different ablation experiments about the weighting factors of the two loss terms are conducted: constant (here, Section 4.2.5) and adaptive (Section 4.2.6) loss weights.

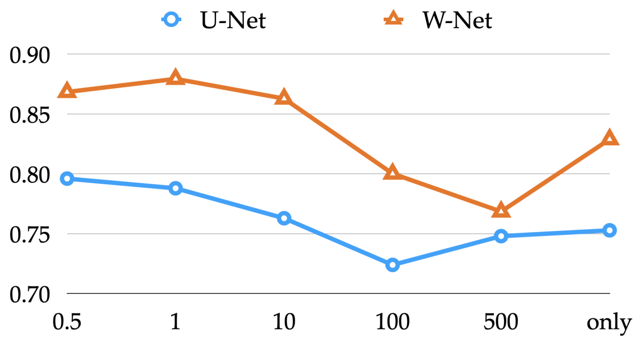
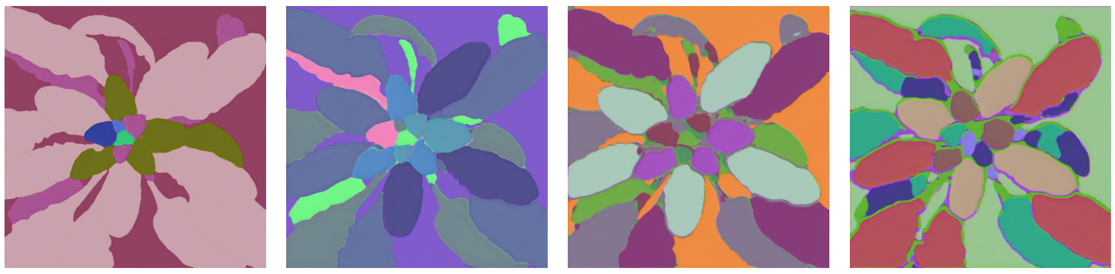


Figure 4.10: Results of Different Loss Weights.
X-axis: weighting factor λ of $\mathcal{L}_{inter}$ as formularized in Eq. 4.3. "only" denotes that the embedding loss depends only on $\mathcal{L}_{inter}$.
Y-axis: mean best Dice scores of testing set.

Implementation

As introduced before, the weighting factor of $\mathcal{L}_{inter}$ is investigated, by slicing it from 0.5 to 500 with keeping the weighting factor of $\mathcal{L}_{intra}$ identical to 1. The only weighting factor that fewer than 1 is chosen as 0.5, to investigate if the amplification of within-instance loss can bring improvement, despite that this breaks the hypothesis. The maximal factor is set up to 500, which seems to be too large. The choice is based on the fact that the embeddings are averaged in the between-instance loss. To reconstruct the supervision from loss gradient upon each individual embedding, the amplification is assumed to be as large as the number of embeddings inside of one cluster. This assumption is very preliminary, but it is still worth being experimented. In addition, an extra experiment termed "only" has also been conducted, where the within-instance loss term $\mathcal{L}_{intra}$ is omitted, formularized as $\mathcal{L}_{emb} = \mathcal{L}_{inter}$. This set-up provides a baseline of the behaviors of two loss terms and baseline of the numerical values of $\mathcal{L}_{intra}$ if the between-instance loss is the only objective. Needless to say that if the between-instance loss $\mathcal{L}_{inter}$ is omitted, the learned embeddings of whole images are all identical.

To sum up, the weighting factors of between-instance loss are `(0.5, 1, 10, 100, 500, "only")`. In the experiments, two network architectures are used: U-Net with 2 Heads and W-Net with Intermediate Supervision and concatenative layer of feature maps for distance regression. In the following context, they are abbreviated as U-Net and W-Net for brevity. Both U-Net and W-Net experiments are using 8 dim embeddings and Polar Form of embedding loss with local constraints. The distance regression is trained with ReLU+MSE loss.



(a) $\lambda = 0.5$ **(b)** $\lambda = 10$ **(c)** $\lambda = 100$ **(d)** only $\mathcal{L}_{inter}$

Figure 4.11: Learned Embeddings with Different Weights λ as in $\mathcal{L}_{emb} = \lambda \cdot \mathcal{L}_{inter} + \mathcal{L}_{intra}$. With ascending λ, overall segmentation performance becomes worse (Figure 4.10) with the decreased consistency of embeddings in the same object. It is worth noting that training with just the between-instance loss term can also to some extent form clusters in the embedding space.

Results

The mean best Dice scores of testing set are illustrated in Figure 4.10. It is clear that the W-Net outperforms U-Net architecture for all weighting factors. For W-Net, the best performance can be obtained by setting constant 1:1 weighting factors, while for U-Net, the halved between-instance class surprisingly brings better results than others. It is prominent that different weighting factors cause significantly different results.

As the primitive of motive to set weighting factors for two loss terms is to emphasize the between-instance loss, the relationship of two terms are worth being investigated apart from the final scores. In Figure 4.12, the ratios of two loss terms $\mathcal{L}_{inter}/\mathcal{L}_{intra}$ are demonstrated with respect to the training steps. It is clear that the larger the weighting factor applied on the between-instance loss $\mathcal{L}_{inter}$ the smaller the loss values are resulted. For the case weighting factor $\lambda = 10$ the numerical values of two loss terms are approximately identical, which however brings in suboptimal performance.

4.2.6 Adaptive vs. Constant Loss Weights

As in Section 4.2.5 experienced, setting a globally constant weight to one of the pair similarities contributes to the performance indistinctively. Inspired by [26], the adaptive weighting factors might regulate the dynamics of the two similarities more spontaneously, as they can be defined in a self-paced manner with respect to the simultaneous losses. The first proposal is defined as below:

$$\mathcal{L}_{emb} = [\hat{\mathcal{L}}_{inter} + m]_+ \cdot \mathcal{L}_{inter} + [\hat{\mathcal{L}}_{intra} + m]_+ \cdot \mathcal{L}_{intra} \, , \tag{4.4}$$

where m denotes the relaxation factor, which ensures the weighting factors for loss terms are not 0 even if they have been perfectly converged, and $\hat{\mathcal{L}}$ denotes the loss

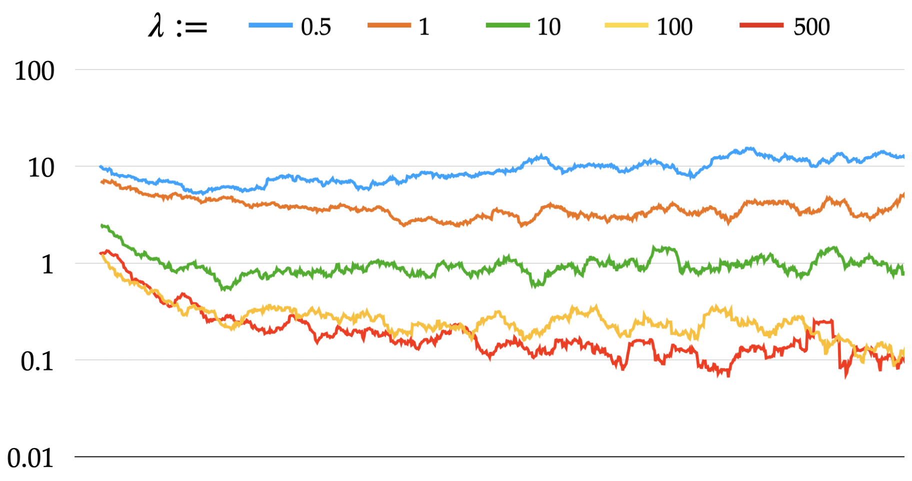


Figure 4.12: Comparison of Constant Loss Weights (with fixed weight of $\mathcal{L}_{intra}$).
X-axis: training steps 0-200k;
Y-axis: ratio of $\mathcal{L}_{inter}/\mathcal{L}_{intra}$ in logarithmic scale.

value that is regarded as a constant that does not participate in the gradient descent progress, implemented via the TensorFlow function `stop_gradient()` [75]. It makes the two terms have the maximal amplification of $1+m$ and shrink of m, where m should be greater than 0 but not much. This modification ensures that the larger values of $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ would have larger weighting factors, and vice versa. In the following discussion, this proposal is called Adaptive v1.

The previously introduced modification has one problem, that the definition of "large" values are absolute, without considering the other term. This might slow down the training progress, if both of the losses are equally small and the learning rate also has been decayed to a small value. To keep the learning rate as the single variable for the step size of gradient descent, the following adaptive weights are presented, which is based on the ratio of the two terms $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$:

$$\begin{aligned} \omega_1 &= \hat{\mathcal{L}}_{inter}/\hat{\mathcal{L}}_{intra} \\ \omega_2 &= \hat{\mathcal{L}}_{intra}/\hat{\mathcal{L}}_{inter} \\ \mathcal{L}_{emb} &= [\mathcal{K}(\omega_1)]_+ \cdot \mathcal{L}_{inter} + [\mathcal{K}(\omega_2)]_+ \cdot \mathcal{L}_{intra} \, , \end{aligned} \tag{4.5}$$

where ω denotes ratio between $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$, $\mathcal{K}(\omega)$ defines a function mapping ratios to ranged weighting factors. $\mathcal{K}(\omega)$ should let weighting factors equal 1, if $\mathcal{L}_{inter}$ and $\mathcal{L}_{intra}$ are equal. And the mapped weighting factors should be ranged, which should also avoid extremely large margins between two terms. Thus two criteria are:

$$\begin{cases} \mathcal{K}(1) = 1 \\ \dfrac{\max\{\mathcal{K}(\omega)\}}{\min\{\mathcal{K}(\omega)\}} \ll \infty \, . \end{cases} \tag{4.6}$$

An example of such a function is depicted in Figure 4.13. Without loss of generality, this is the used function in the following experiment. The second proposal takes the relative ratio of two terms in consideration, which can avoid the case of doubled effect of decreasing the learning rate. It is called Adaptive v2 in the following discussion.

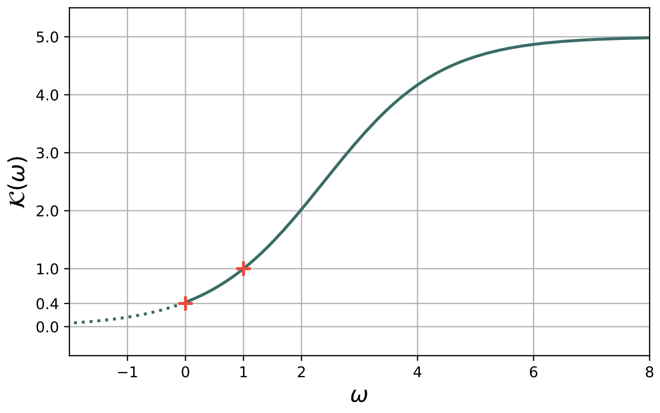
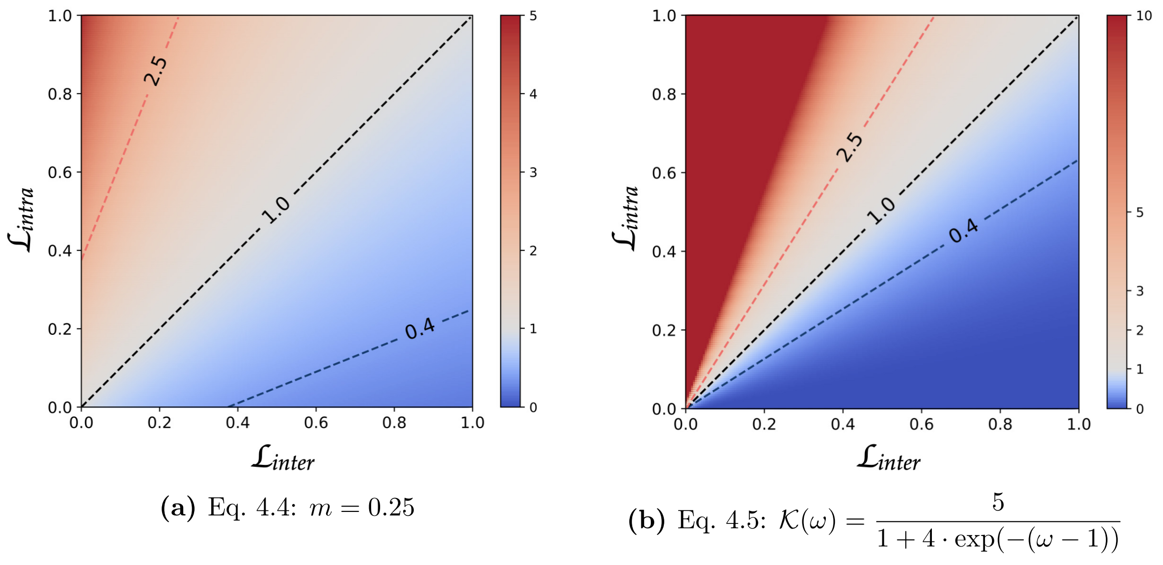


Figure 4.13: An Example: $\mathcal{K}(\omega) = \dfrac{5}{1 + 4 \cdot \exp(-(\omega - 1))}$, where $\mathcal{K}(1) = 1$, $\min\{\mathcal{K}\} = \mathcal{K}(0) = 0.4$ and $\max\{\mathcal{K}\} = \mathcal{K}(\infty) = 5$

Implementation

In this ablation experiment, three formats of weighting factors are investigated. The vanilla version is the constant and identical attribution of two terms, termed Constant 1:1. The first adaptive weighting format is according to Equation 4.4 with $m = 0.25$, termed Adaptive v1 and the second adaptive weighting format is according to Equation 4.5 with $\mathcal{K}(\omega) = \dfrac{5}{1 + 4 \cdot \exp(-(\omega - 1))}$, termed Adaptive v2.

As in Figure 4.14 shown, both versions of proposed adaptive weighting formats have the characteristic of encouraging the equally valued within-instance and between-instance losses. In addition to the difference mentioned previously, that the weighting factors are 1 if the losses of two terms are equally small for Adaptive v2, there are also other differences. The Adaptive v2 is more radical, according to Figure 4.14, as the theoretical maximal relative amplification is $\dfrac{\max\{\mathcal{K}(\omega)\}}{\min\{\mathcal{K}(\omega)\}} = \dfrac{\mathcal{K}(0)}{\mathcal{K}(\infty)} = 12.5$, while the maximal relative amplification of Adaptive v1 is $1.25/0.25 = 5$. Moreover, the goal of Adaptive v2 is more straightforward to the optimum, as the isolines in Figure 4.14 illustrated.



(a) Eq. 4.4: $m = 0.25$

(b) Eq. 4.5: $\mathcal{K}(\omega) = \dfrac{5}{1 + 4 \cdot \exp(-(\omega - 1))}$

Figure 4.14: Comparison of Adaptive Weights.
Colors denote the ratios of the weighting factors intra/inter.

All three experiments are conducted with the embeddings of 8 dims using W-Net with intermediate supervision of preliminary embedding loss. The concatenative layer is the distance features. The loss functions are in the format of Polar Loss Function and with local constraints.

Results

Table 4.5 shows the results of three set-ups with respect to the mean best Dice scores. The results indicate the adaptive weights for the two loss terms cannot effectively improve the final performance based on the used dataset and parameters. Despite that the results are not expected, the hidden circumstances are worth being investigated.

Table 4.5: Results of Adaptive and Constant Loss Weights.
Evaluated by mean best Dice.

Constant 1:1	Adaptive v1	Adaptive v2
87.9%	86.0%	84.2%

The modification of weighting factors is based on the assumption, that the numerical values of two loss terms are of equivalent meanings. If this assumption applies, the same values of two loss terms should indicate the same degree of how they are optimized. Revisit the loss formats of two terms, the within-instance loss is referred to the distances between points and their centroids, while the between-instance loss is referred to the distances between different instance centroids: the formats are *not symmetric*. Despite that the theoretical upper and lower bounds of two loss terms are identical, namely 1 and 0, the intermediate values may reflect the level of optimization heterogeneously. Due to the asymmetry of the loss formats, this assumption can be hardly turned to be confirmed.

It is still interesting to verify that if the numerical values of two loss terms are controlled as desired, even if the regularized loss values do not imply the expected improvement on final performance. Figure 4.15 depicts the ratios $\hat{\mathcal{L}}_{inter}/\hat{\mathcal{L}}_{intra}$ with respect to the training steps from 0 to 200k. Before 60k steps, the ratios $\hat{\mathcal{L}}_{inter}/\hat{\mathcal{L}}_{intra}$ are of expectation. In the later stage, the effect of adaptive weighting factors disappeared. It can be confirmed from this observation that with the adaptive weights the values of two loss terms are controlled as expected in the early training stage.

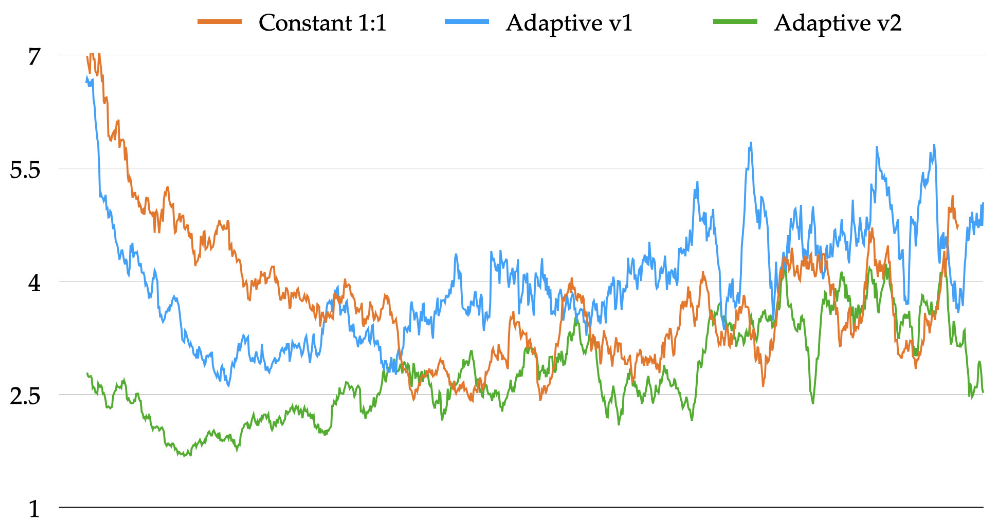


Figure 4.15: Comparison of Adaptive Weights.
X-axis: training steps 0-200k; Y-axis: $\hat{\mathcal{L}}_{inter}/\hat{\mathcal{L}}_{intra}$.
Before 60k steps, the ratios $\hat{\mathcal{L}}_{inter}/\hat{\mathcal{L}}_{intra}$ are of expectation. In the later stage, the effect of adaptive weighting factors disappeared.

4.2.7 Dimension of Embeddings

As mentioned in Section 2.2.3, if the embeddings with local constraints could be perfectly and efficiently learned, only four labels would be generated. With this extreme case and the benefits of local constraints in mind, the dimension of embeddings should be set low. In [13] and [10], the dimension of embeddings is set to 32 and 16 respectively to achieve best performance. Furthermore, it has been compared using 4 dimensional and 16 dimensional embeddings in [10]. In this experiment, a wider range of dimensions are investigated, namely 4, 8, 16, 32 and 64 dims. The results showcase that it is beneficial to leverage local constraints in instance segmentation, inasmuch as lower dimensions have advantages regarding with computation and memory overheads.

To avoid confusion, the dimension investigated here is the one of final embeddings. The U-Net backbone is identical for all experiments with the first convolutional layer transforming arbitrarily dimensional vectors to 32 dim and with the last convolutional layers (considered as a block, as the last convolutional layer does not change dimension) transforming back to 32 dim. The dimension discussed in this experiment is referred to the settings of head-dependent joint block after the U-Net backbone, namely the dimension of outputs of corresponding heads after U-Net backbones. See Appendix A for more details of network architectures.

Implementation

Without loss of generality, the embeddings of intermediate supervision, the final embeddings and the feature maps of distmap head are set identically from 4 dim to 64 dim. Two network architectures are taken in this experiments: U-Net with 2 heads 4.7(b) and W-Net with supervision 4.7(d). In the following discussion, they are abbreviated by U-Net and W-Net for brevity. The concatenative layer is the feature map of the distmap head in the end of the first U-Net. All feature maps are of 32 dimensions after the first convolutional layer. The loss of distance regression is ReLU+MSE. As the higher dimension requires higher GPU memory, all the experiments up to and including 16 dim are trained with `batch_size=4`; 32 dim and 64 dim are trained with `batch_size=2` to decrease the memory consumption.

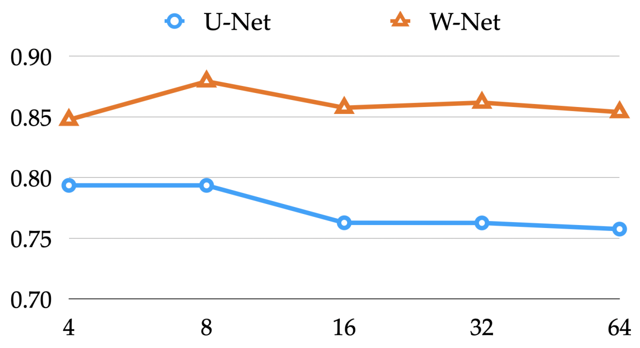


Figure 4.16: Results of Different Embedding Dimensions.
X-axis: the number of dimensions of embeddings in the loss functions.
Y-axis: mean best Dice scores of testing set.

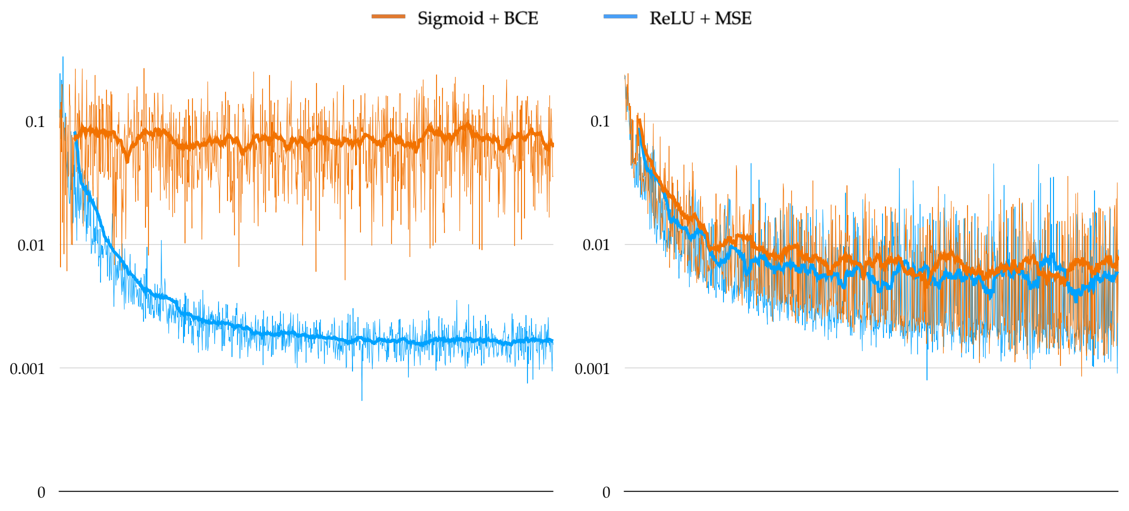
Results

The mean Symmetric Best Dice (mSBD) of CodaLab's submission is the evaluation metric. Figure 4.16 shows the results. Not surprisingly, W-Net outperforms U-Net in all dims. In the cases of U-Net, the results are almost equally better for 4 and 8 dim, reaching around 79%, than higher dimensions, around 76%. In contrast, 4 dim results in 85% with W-Net, which is surpassed by the best performance of 88% for 8 dim.

The higher the dimensions are, the more information and features the embeddings may contain. It is interesting to notice that the higher dimensions do not mean the better performance. And it saturates at 8 dim, which is the chosen parameter in the final methodology. This brings the best mSBD scores with the lowest possible dimensions of embeddings.

4.2.8 Loss for Distance Regression

Different loss functions may have tremendous influence upon the results. Some of them can be explained only intuitively or empirically, as the complicated training progress is quite difficult to be proven precisely. In this task, the format of loss function for distance regression is one of the examples. The target ground truth is ranges between 0 and 1, thus the most intuitive choice is the Linear+MSE, which has an adequate interpretation as maximizing likelihood with respect to probability theory [76]. The replacement of linear activation function with ReLU is natural, as only non-negative values are expected. In contrast, Sigmoid+BCE seems to be an alternative, as the output of the combination of Sigmoid+BCE is continuously ranged between 0 and 1, which fulfills the desired outputs. In the following experiment, the real-world results are compared with these two set-ups.



(b) Losses of Distmaps **(c)** Losses of Embeddings

Figure 4.17: Loss Comparison for Distance Regression.
X-axis: training steps 0-200k; Y-axis: loss in logarithmic scale.
The bold curves represent moving averages of original values.

Implementation

The W-Net with intermediate supervision network is used in this experiment, where the first U-Net outputs 8 dimensional feature maps and the second 16 dimensional. Concatenative layer is the feature map directly from last layer of the first U-Net, with the shape of $(512, 512, 8)$. The distance regression loss is added after the head for distance feature map. In this experiment the two types of activation functions and loss functions are compared, namely ReLU+MSE and Sigmoid+BCE. The formulae are introduced in Section 2.5 via Equation 2.13 - Equation 2.17.

Results

The following three points are evaluated and compared:

- the losses of distance regression with respect to training steps;
- the losses of final embeddings with respect to training steps;
- the final best dice scores.

Figure 4.17 shows the losses of distmaps and embeddings respectively. Figure 4.17(b) indicates that the Sigmoid+BCE is hard for distance regression to converge or extremely easy to saturate, whereas the combination of ReLU+MSE has a clear convergence during training. Despite that conspicuous difference, the losses of embeddings are not as much influenced. In Figure 4.17(c), the orange curve is more often located above the blue one, which can be corroborated by final best dice scores: ReLU+MSE outperformed Sigmoid+BCE by 1.6% from 84.2% to 85.8%.

4.2.9 Clustering

Apart from the default angular clustering used along through the experiments, other three clustering techniques have been tested based on the predicted embeddings of best results: Mutex Watershed [39], Mean Shift [30] and HDBSCAN [37]. On the one hand, this provides a reference for the performance of different clustering methods on the embeddings trained with cosine similarity based loss. On the other hand, it can also indirectly reflect the quality of embeddings generated by U-Net and W-Net. Results are shown in Table 4.6.

Results

In conclusion, the angular clustering has advantages in terms of performance and speed. Nevertheless, it should be noted that this method is only applicable to the case, where seeds are available and clusters are orthogonal in the embedding space. Additionally, all clustering approaches produce better results with embeddings predicted from the W-Net, which again confirms the improvement of our proposed method.

Table 4.6: Comparison of Local/Global Constraints, Network and Clustering. Denotations: Local = local constraints, otherwise global; 64d = 64 dim embeddings, otherwise 8 dim; AC = Angular clustering; MWS = Mutex Watershed [39].

Local	Net	Clustering	mSBD
✓	W-Net	AC	**.879**
✓	W-Net 64d	AC	.854
	W-Net	AC	.835
	W-Net 64d	AC	.823
✓	U-Net	MWS	.719
✓	W-Net	MWS	.771
✓	U-Net	MeanShift	.679
✓	W-Net	MeanShift	.733
✓	U-Net	HDBSCAN	.631
✓	W-Net	HDBSCAN	.681

4.3 Comparison against State-of-the-Art

Comparison of state-of-the-art methods on the CVPPP LSC dataset is quantitatively shown in Table 4.7.

It is clear that the learning based methods (denoted with backbones) can achieve better results than the first four classical methods (IPK [4, 60], Nottingham [4], MSU [4,62] and Wageningen [4]). The last four methods (DiscLoss [11], CE-RH [13], E-LC [10], W-Net [77]) are based on embedding learning with similarity pair loss. Roughly speaking, they bring in promising results. The overall result mSBD for A1-5 outperforms all others. In the leaderboard, our overall result is at the 1. position by paper submission. Furthermore, the average of mSBD scores for Arabidopsis images (A1, A2, A4) outperforms the second best results from three different users respectively by over 4%, namely 0.873 to 0.917. Due to the extremely imbalanced training images on Arabidopsis (783 images) and Tobacco (27 images), the results on testing set A3 are not as good as others, with mSBD of 0.77. Compared to this, the current 1. place mSBD of A3 in the leaderboard reaches 0.89. It implies that the sufficient number of training images is critical in our proposed method. We leave this room for improvement in the future. More discussions about the proposed method can be found in Chapter 5.

One thing worth mentioning is that the authors tend to not submit their results to the leaderboard of CodaLab, which makes the consistent comparison and review rather difficult.

Table 4.7: Comparison of Results. Abbreviations: Aug. = Data augmentation; Emb. = Metric of embedding similarity; Fg. = Ground truth foreground masks are used; syn = Synthetic images are used for training; HG = Stacked Hourglass network; Lb. = Results shown in the leaderboard of CodaLab.

Method	Backbone	Train	Aug.	Emb.	Fg.	Lb.	mSBD		
							A1	A1-3	A1-5
IPK [4,60]	-	A1-3			✓		.791	.782	-
Nottingham [4]	-	A1-3			✓		.710	.686	-
MSU [4,62]	-	A1-3			✓		.785	.780	-
Wageningen [4]	-	A1-3			✓		.773	.769	-
MRCNN [10,42]	ResNet	A1-3					-	.797	-
Stardist [10,15]	U-Net	A1-3					-	.802	-
IS-RA [78]	FCN	A1					.849	-	-
Ward [65]	ResNet	A1-4+syn	✓				.900	.740	.810
UPGen [66]	ResNet	A1-4+syn	✓				.890	**.877**	.874
DiscLoss [11]	ResNet	A1	✓	euc	✓		.842	-	-
CE-RH [13]	HG	A1	✓	cos			.845	-	-
E-LC [10]	U-Net	A1-3		cos			-	.831	.823
W-Net (ours)	U-Net	A1-4		cos	✓	✓	**.919**	.870	**.879**

4.4 Application on Human U2OS Cells

The proposed method has also been tested on the image set BBBC006v1 of human U2OS cells from the Broad Bioimage Benchmark Collection [7]. Totally 754 images are randomly separated into two equally distributed training (including 20% validation set) and testing set with 377 images respectively. The images are of single channel and the data type is `uint16` instead of `uint8` as in CVPPP Leaf Segmentation dataset. Other set-ups are identical to previously introduced ones. U-Net and W-Net with distance concatenative layer have been used to show results with mSBD and mean Average Precision with IoU={0.5, 0.55, 0.6, ..., 0.9} (mAP). The definition can be found in Section 2.5.

4.4.1 Results

The mSBD has increased from 0.896 to 0.915 and the mAP from 0.577 to 0.664. The examples of final labels on the testing set are illustrated in Figure 4.18. As reported in [10], some embeddings around boundaries might be incomplete, which leads to incomplete segmentations. This problem has been mainly solved as showcased in Figure 4.18. Nevertheless, there are still objects that occasionally cannot be detected. The post-processing can also be further finetuned.

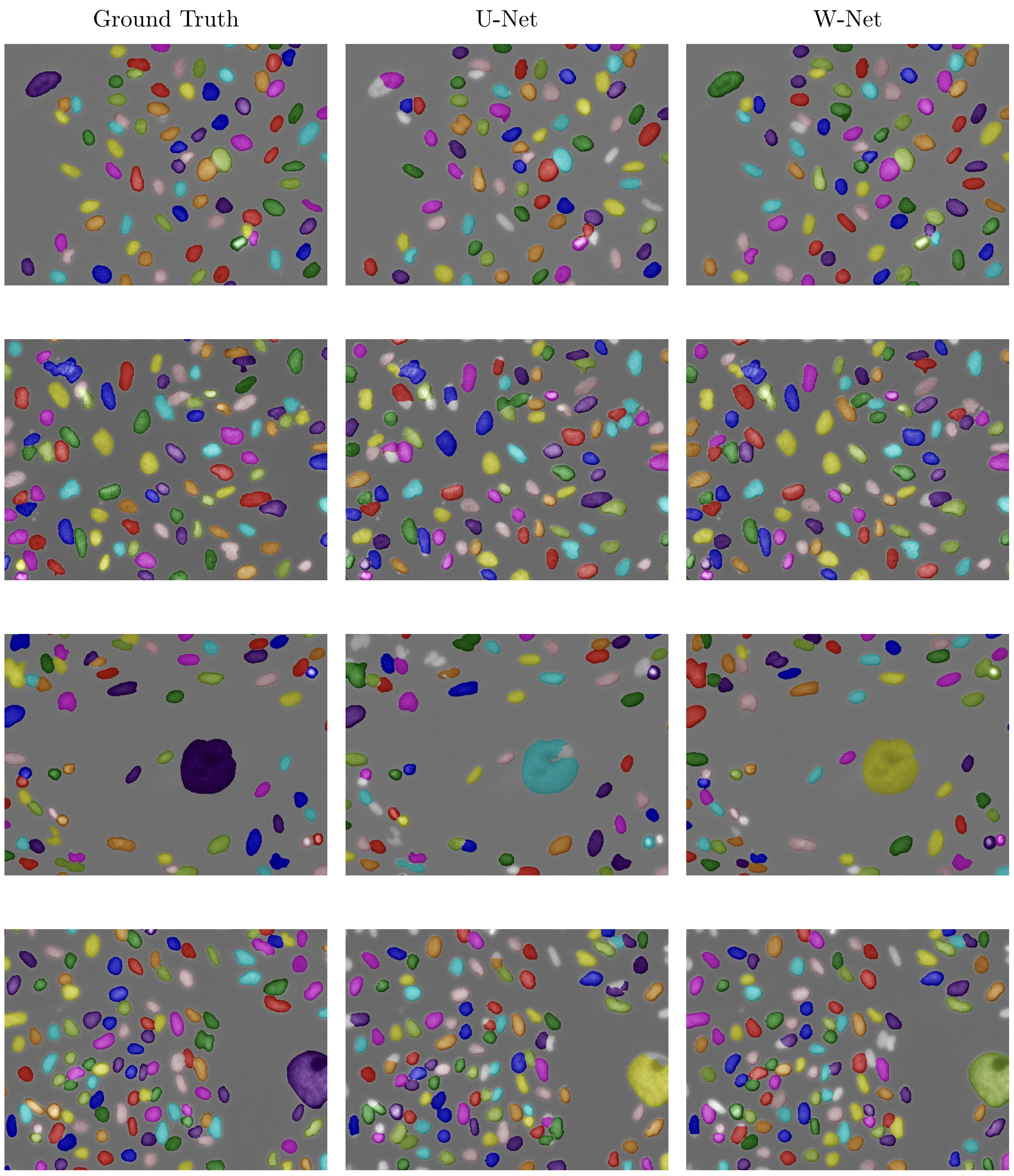


Figure 4.18: Cell Segmentation Results of the BBBC006v1 Data: Ground Truth (left), U-Net (middle) and W-Net (right); Each row illustrates one example. Improvement from U-Net to W-Net is salient, as on the CVPPP LSC dataset. The mSBD and mAP score have increased from 0.896 to 0.915 and from 0.577 to 0.664, respectively. The learned results of W-Net are far from optimal: some cells are occasionally not detected. The fourth row illustrates one mainly failed example.

4.5 Application on Cityscapes

Lastly, the proposed method is tested on the multi-class dataset: Cityscapes. The experiment is considered as a toy example to confirm the feasibility of the proposed method on the more complicated dataset. In the following experiment, the classes of objects are ignored, i.e., each object is considered as a single instance without mentioning it is a car or a person. It is more challenging, as the network should learn the more general features amongst all semantic classes. Furthermore, the results are only visually illustrated, without giving the evaluation scores quantitatively. The reason is that this thesis if focusing on the instance segmentation, and the quantitative evaluation of multi-class instance segmentation requires semantic labels in advance. The results have shown decent accuracy of segmentations even without ad-hoc refinement, which inspires the future work to expand the proposed method to an end-to-end instance segmentation method for multi-class dataset.

The most salient difference of implementation is the augmentation, which is introduced in Section 4.5.1. Following that, a plenty of examples of instance segmentation predictions are visually illustrated.

4.5.1 Augmentation

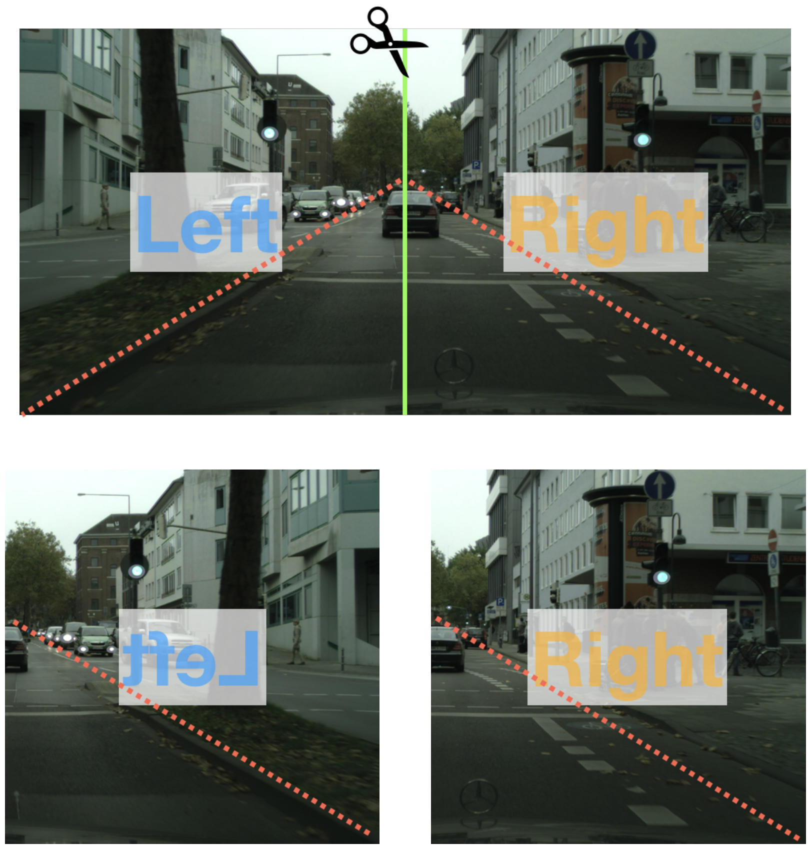


Figure 4.19: Augmentation for Cityscapes.

As the image size (1024×2048) of Cityscapes dataset is larger than Leaf Segmentation and the images are rectangular, not square formed as in Leaf Segmentation, the simple augmentation is applied to the images before training. As depicted in Figure 4.19, the images are equally cut to two parts firstly. Then, the left part is horizontally mirrored, since all images are structurally symmetric. Finally, both mirrored left part and original right part are resized from (1024×1024) to (512×512) and are fed into the W-Net.

This augmentation can bring two advantages:

- The size of each input image is decreased from (1024×2048) to (512×512), which enables training using batch on a single GPU;
- As denoted in Figure 4.19, the perspectives from left and right parts are unified. The training images then have the vanishing point always at the middle of the left boundaries, which can promote the robustness of training processing.

4.5.2 Visual Results

In this toy experiment, only visual results of predictions on the testing set are illustrated qualitatively in Figure 4.20 and Figure 4.21. The results show that the proposed method has promising potential also on multi-class instance segmentation. Even the toy experiment has ignored the semantic classes, it results in still decent segmentations.

In the testing images with different classes, like car and person, the network is capable to detect instance objects of all classes. The adjacent objects can also be to some extent correctly segmented. Even in some abnormal cases, like a person before a tram, both objects of extremely different sizes can still be detected and segmented.

The problems are also salient. Occasionally, the adjacent objects cannot be segmented, same as the experiments in other datasets. Moreover, some objects cannot be detected. The possible reason might be the poor quality of distance regression, as in this case, it is much more difficult to train due to the more complicated background and multiple semantic classes.

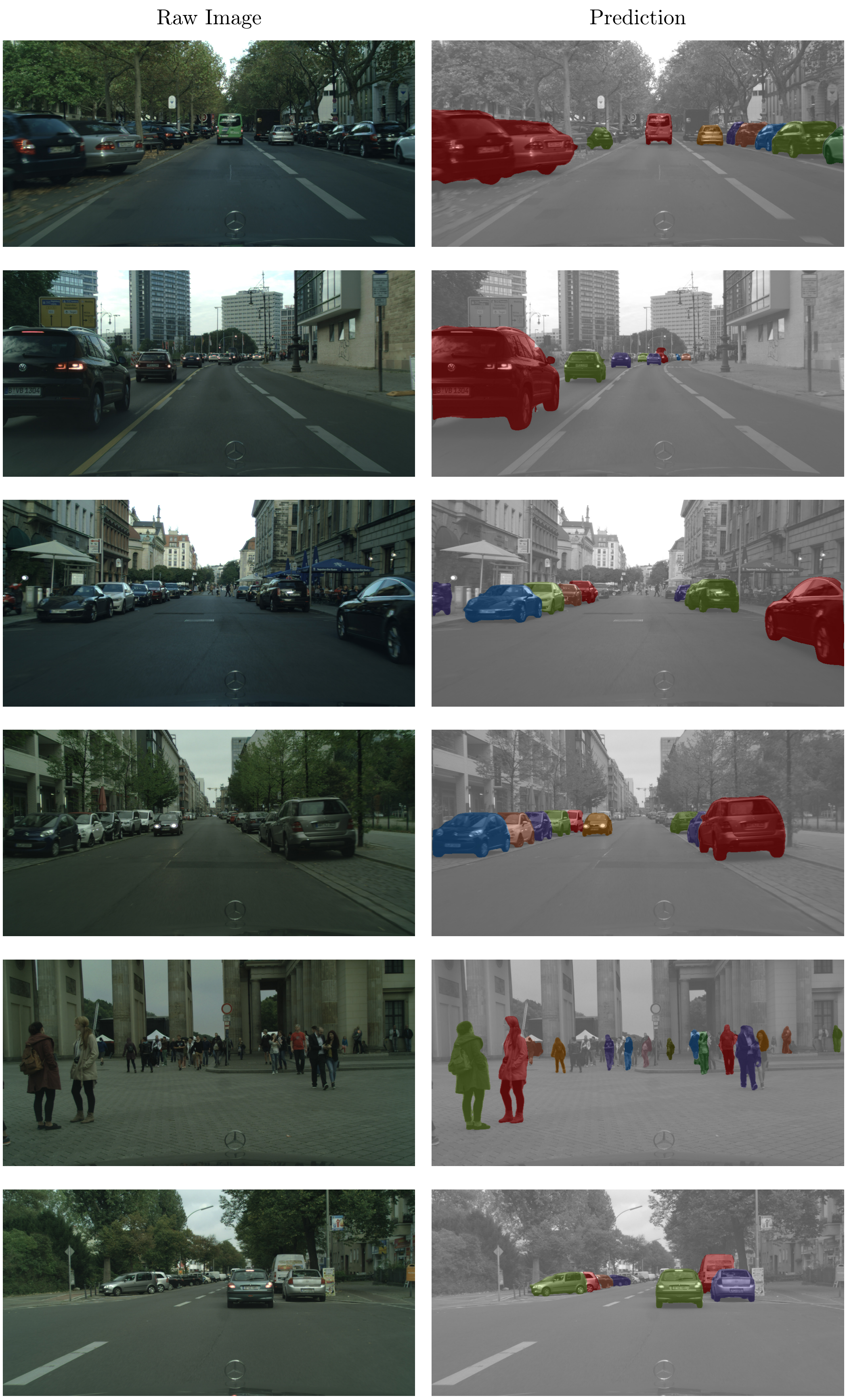


Figure 4.20: Citycapes Results: Part 1.

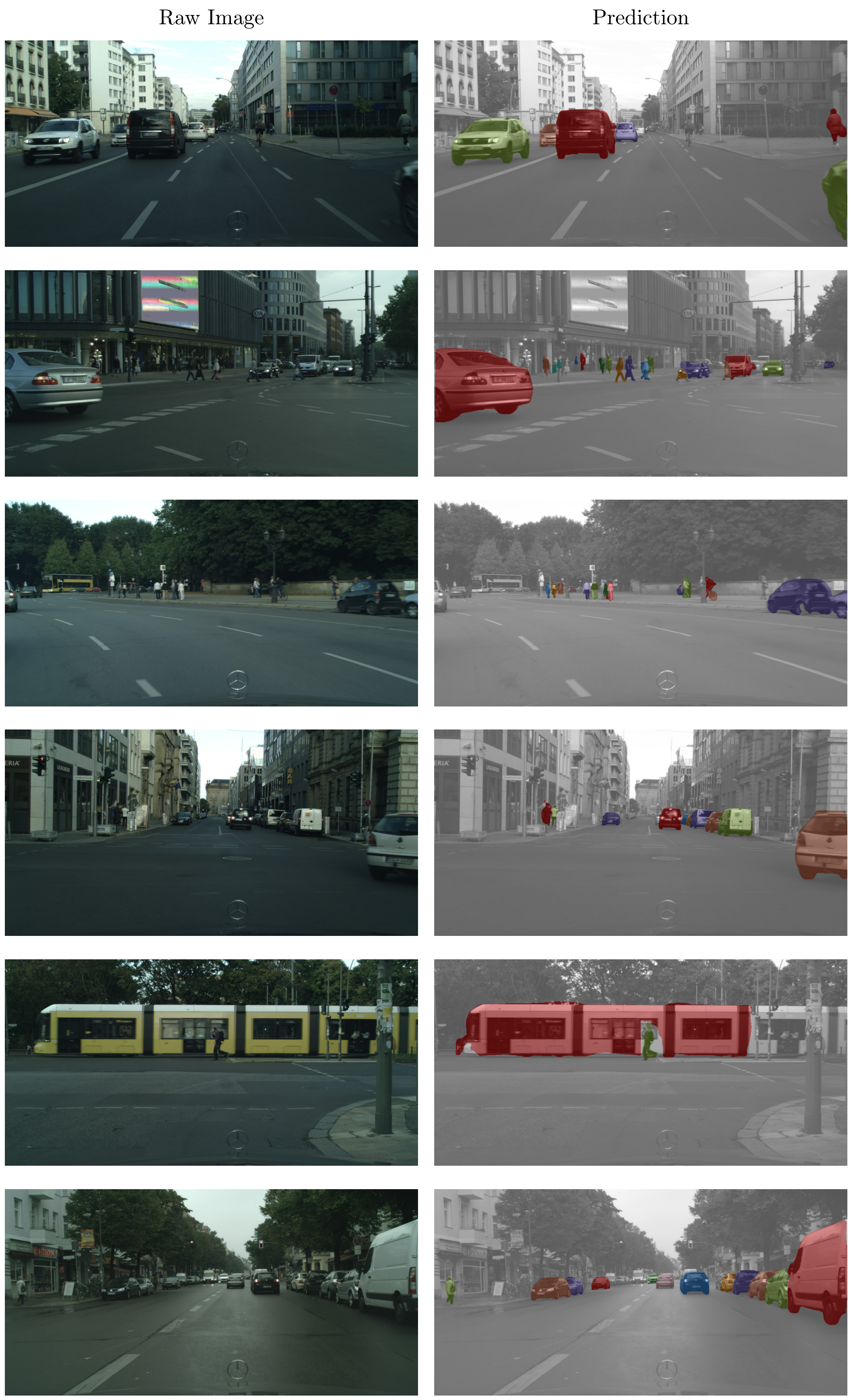


Figure 4.21: Citycapes Results: Part 2.

5 Evaluation

To this end, the improvement made by the proposed W-Net has been showcased in the previous chapters. In this chapter, the proposed method is evaluated primarily with respect to the shortcomings. The failed results are discussed and the corresponding weaknesses of the proposed method are presented in Section 5.1. Following that, the widely used format of similarity loss pair is revisited in Section 5.2.

5.1 Weaknesses

The proposed method has several noticeable weaknesses. First, it often fails in the cases when the instance objects are extremely overlapped. Moreover, although it is able to promote the accuracy of segmentation, the overhead of computation is considerable. It is a detour compared to the original U-Net with two heads, as the shared top-down pathway ensures the efficient training and testing progress. The trend of deep learning network construction shall be the shared features and parallel architectures rather than serial architectures. The proposed method has achieved the improvement of accuracy but lost the elegant architecture and efficiency. This can be compensated by, for example, subjoining semantic segmentation network into the first distance regression module. Last but not least, the method is finetuned upon the relatively uncomplicated dataset CVPPP LSC, which contains only one class. The distmap is easy to train, inasmuch as the instance objects are mainly of the same color: green. The application on other more complex datasets, such as Cityscapes, needs more detailed refinement. In the following sections, these shortcoming are investigated in details.

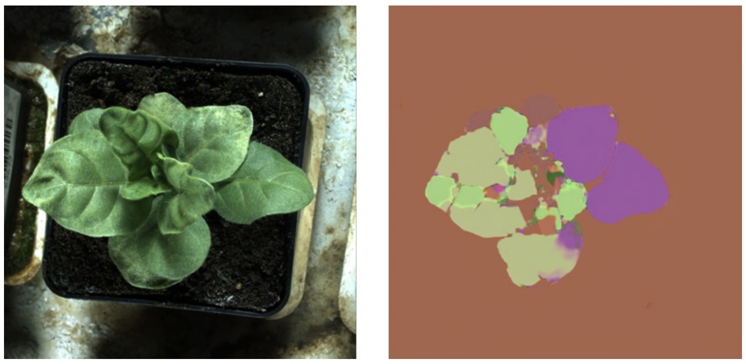
Table 5.1: Results of Testing Sets. Evaluated by mean Symmetric Best Dice. The proposed method results in the best performance amongst all testing sets except for Tobacco (A3).

Leaderboard	A1	A2	A3 Tobacco	A4	A5	A1-5	A1,2,4 Arabidopsis
1. Place	92%	92%	89%	91%	88%	88%	92%
2. Place	92%	86%	88%	87%	88%	88%	88%
Proposed	**92%**	**92%**	77%	**91%**	**88%**	**88%**	**92%**

5.1.1 Demanding Training Images

The detailed experiment results are shown in Table 5.1 per testing set. The testing sets consist of 5 files A1-5, where A1, A2, A4 contain images of Arabidopsis, A3 contains images of Tobacco, and A5 is the mixed file from previous four files. The testing sets contain {33, 9, 56, 168, 235} images for A1-5, respectively. All images from A3 appear also in A5, i.e. the number of testing images for Tobacco is 112. In the contrast, the training sets consist of 4 files A1-4, where A1, A2, A4 contain totally 783 images of Arabidopsis, and A3 contains only 27 images of Tobacco. Results from first and second place in the leaderboard at CodaLab together with the proposed results are compared.

It is salient that the only disadvantage of the proposed method is caused by Tobacco images. No doubt that the extremely imbalanced training and testing sets play a significant role in the performance. Apart from this, the results also show that the sufficient number of training images is desirable in the proposed method. The second possible reason is that the Tobacco images are, to some extent, more complicated, as the secondary veins are also rather salient. This makes it even more difficult to distinguish the veins inside of leaves and the boundaries between leaves. As a result, the large leaf is often segmented to several smaller parts, as shown in Figure 5.1 (a).


(a) Failed Case of Tobacco Image

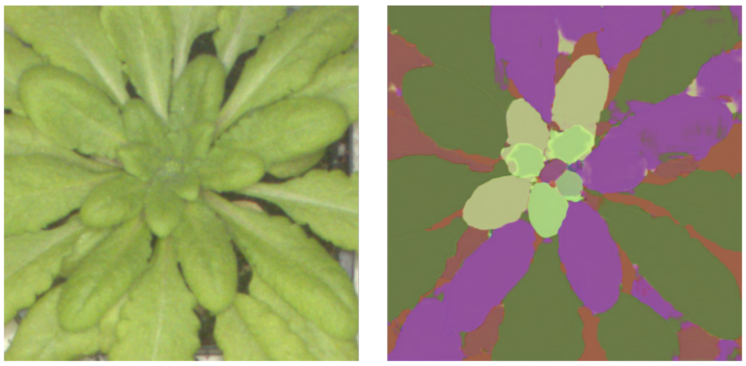

(b) Failed Case of Large Leaves

Figure 5.1: Failed Cases.

5.1.2 Scale Inflexibility

The dataset of CVPPP Leaf Segmentation Challenge consists of the plants through out their growing periods, from tiny sprouts to large leaves. In the experiments, it is found that those plants with extremely large leaves can be hardly decently segmented. One example is shown in Figure 5.1 (b). Three facts can be the potential reasons for the phenomenon:

1. The number of training images of this type is insufficient. It is similar for the Tobacco training set;
2. The number of instances in one image is large, which might make the embedding space more difficult to train;
3. Last but not least, they are tough problems themselves. The leaves are severely overlapped, which leads to the circumstance that a number of them are incomplete. Furthermore, the ambiguity between midveins and boundaries is more critical.

Similar cases can also be found from the results on Cityscapes (Figure 4.20 and Figure 4.21), where the large objects and small objects are the most vulnerable parts.

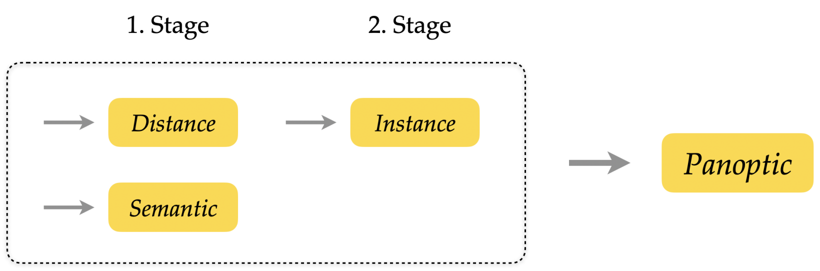


Figure 5.2: Expected Processing Pipeline towards Panoptic.

5.1.3 A Detour for Accuracy

One of the trends of novel network architectures is to compress the ad-hoc heads based on the shared architectures and to exploit the shared features as efficiently as possible. The introduced U-Net with two heads is one of the examples. The benefit is clear, that the computational overhead can be significantly reduced. The proposed W-Net does improve the accuracy of segmentation, however, it breaks up the elegant design for more efficiency. It is considered as a temporary detour between the single-class instance segmentation and general instance segmentation. The first stage can be subjoined with other tasks, such as semantic segmentation. The whole processing pipeline can thus be depicted as in Figure 5.2, making it towards panoptic. To this end, each pixel of original images is mapped to a higher-dimensional embedding with both semantic labels and instance labels. In this outlook, the first stage of distance regression can be performed more efficiently, as the features can also be shared to the semantic segmentation task.

5.1.4 Application Constraints

The core of the proposed method is to exploit the features of distance regression, which implies that the distance regression shall be easy to learn to achieve the improvement. Intuitively speaking, it is believed that the distance regression is generally easier to learn, compared with embedding module. Nevertheless, the salient improvement can be seen, if the dataset is single-class and if the target objects are similar. The CVPPP LSC is a perfect example. The detailed finetuning for more complicated datasets, such as Cityscapes, is expected, to show the feasible application of the proposed method on general towards real-world cases. Conclusively, the proposed method is currently preferably suitable for single-class instance segmentation, where the foreground and background are relatively easy to recognize.

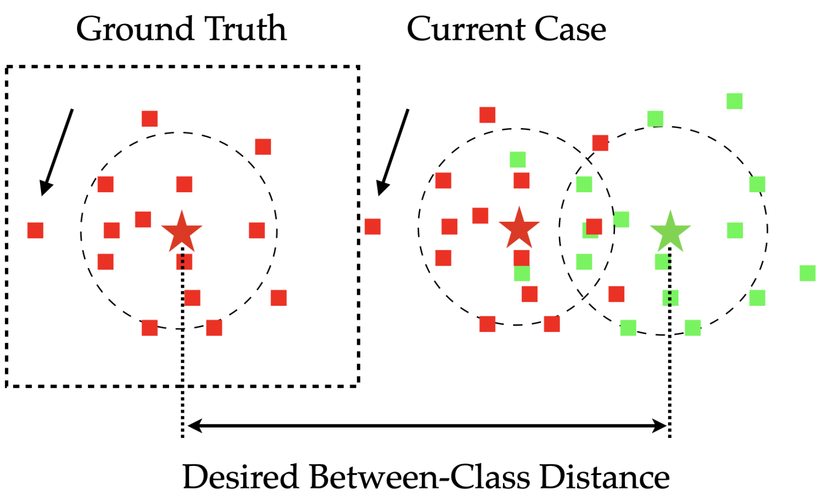


Figure 5.3: Dilemma of Embeddings: Left or Right?

5.2 Rethinking Similarity Loss Pair

The loss format consisting of two similarity terms is widely used in the approaches of pixel embedding learning. It shows potential advantages in handling complex shapes and dense objects. However, in some specific circumstance, this loss format could be imperfect, as the guidance from the two loss terms might be contradictory. Consider a toy example as illustrated in Figure 5.3. There are two clusters: red and green, with the green embeddings being fixed for brevity. The tiny rectangles denote the embeddings and the stars denote the cluster centroids. The ground truth status of red cluster is depicted with a dotted rectangle, in which case the desired between-class distance from the red cluster centroid to the green is also illustrated. The right clusters of red and green denote the current case. Take a closer look at the arrowed red embedding:

- The within-instance loss term encourages this embedding to move closer to the red centroid (marked as star): the *right* direction;
- The between-instance loss term expects longer distance between two clusters, which results in the ground truth status as the left half image illustrated. In this context, the embedding seems to be pushed to the *left* direction.

To sum up, the arrowed embedding has double issues: it is confused by the between-instance and within-instance terms at the same time. Taking this toy example into consideration, it is believed that this loss format is suboptimal, as in some specific circumstance, the guidance from the loss function can be contradictory. It is a strict criticism, as this unstable situation can be mitigated if the embeddings are either pushed together first or pulled away first. After that, the loss will focus on the other problem. Or the numerical values of each term can reflect the priority for the upcoming action.

It is believed that the whole training progress with neural convolutional layers under this loss format is so *result-oriented*, that the hidden processing can hardly be observed or controlled. [79] has proposed a mechanism for generalization of 3D point clouds spatial relations that can perform optimization beyond convolution. I believe that this kind of module could inspire to design a novel mechanism that could impose direct supervision on the transformation from pixels to embeddings.

6 Final Remarks

6.1 Conclusions

There are two main contributions of this thesis. The first one is corresponding to the novel W-Net architecture, which promotes the single-class instance segmentation significantly. The second one is corresponding to the thorough analysis of U-Net based cosine embedding learning approach with local constraints. A number of ablation experiments have been conducted to supplement the vacancies in this field. In the following, these two contributions are conclusively presented.

The baseline of this thesis is the U-Net with two heads in [10], where the pixel embedding learning using cosine similarity and local constraints are used. The results of the application on CVPPP Leaf Segmentation Challenge were on par with the state-of-the-arts, however, some adjacent objects were not correctly segmented occasionally. This is the motivation of this thesis: to improve the performance on the basis of the U-Net based pixel embedding learning using cosine similarity and local constraints on single-class dense instance segmentation.

During the experiments, it is found that the distance regression is easy to learn, both accurately and fast. The distance regression map can give the cues of the location of objects (object-ness) and the clear differences of the inside regions and the boundaries of the objects. To this end, it is nature to exploit the distance regression results more thoroughly to achieve the improvement of segmentation accuracy. They are concatenated to the images as the new inputs for the other U-Net, as it is believed that the concatenation can reserve more information than addition, although the overheads of computation and memory are accompanied. Following that, a number of ablation experiments about this concatenative layer are conducted. The conclusions are demonstrated in Section 6.1.1

The second contribution is regarded with the thorough ablation experiments on the basis of pixel embedding learning, covering the cosine similarity and Euclidean similarity metrics, the local and global constraints, the loss functions of distance regression, the different dimensions of embeddings and the constant and adaptive loss weights. In advance of this work, relative studies are mostly vacant to my best knowledge. The proposed work aims to indicate the potential influences of these parameters on the training processing and the final performance. The conclusions are demonstrated in Section 6.1.2.

6.1.1 Concatenative Layer

The main novelty of this work is the exploit of intermediate distance regression supervision: the distance concatenative layer. A set of candidates of concatenative layer are investigated, including the 1-dimensional distmap, 8-dimensional distance features, 32-dimensional distance features, 32-dimensional embedding features, concatenated 16-dimensional distance features and 16-dimensional embedding features. The cases of using embedding features are the application of stacked networks with intermediate supervision [19].

The results show that the stacked network with intermediate embedding supervision can improve the performance, so can those with distance feature supervision even further. In conclusion, the proposed distance concatenative layer is able to promote the segmentation accuracy significantly with minimal computation overhead. It is believed that this module can also be subjoined with other state-of-the-art network architectures. Moreover, it is expected to apply the proposed method to more datasets.

6.1.2 Pixel Embedding Learning

A number of ablation experiments have been conducted corresponding to the different set-ups of the network architectures and the loss formats. Apart from the types of concatenative layers, the following parameters have been investigated: the cosine similarity and Euclidean similarity metrics, the local and global constraints, the loss functions of distance regression, the different dimensions of embeddings and the constant and adaptive loss weights.

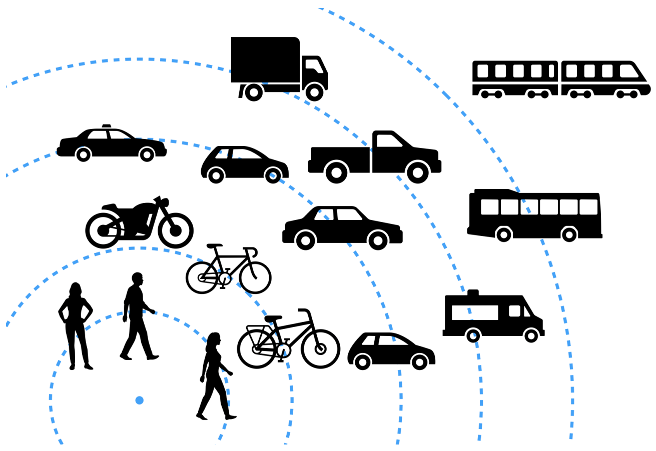


Figure 6.1: Outlook of Embeddings. The radius represents the semantic label (class) and the angle represents instance label.

6.2 Outlooks

6.2.1 End-to-End Multi-class Instance Segmentation

The proposed method is primarily focusing on the single-class instance segmentation, despite that the Cityscapes dataset is also tested as a toy example to show the promising capability. The next step is naturally the end-to-end multi-class instance segmentation. As discussed in Section 5.1.3, the first stage can be subjoined with more complicated tasks, like semantic segmentation. Multi-task image segmentation via deep embedding learning is the desired goal in the future. Each pixel of the original image is mapped to a embedding vector of rich features, containing both semantic information of which given class this pixel belongs to, and instance information of which individual object this pixel belongs to. Figure 6.1 illustrates an example of expected embeddings.

Furthermore, as the embedding approaches have achieved much success in the field of NLP, where word2vec [80] has successfully built the general word representations. The embeddings from images can also be mapped to the same domain as the words in the NLP field, making the embeddings themselves full of semantic meanings.

6.2.2 Similarity Loss Pair

As discussed in Section 5.2, the hidden mechanism behind the similarity loss pair consisting of between-instance loss term and within-instance loss term remains unknown. It is believed that *why* is more important than *how well* for the future research. Therefore, this loss format is very worth being further studied. There are a few differences between different similarity loss proposals: the L1 norm based loss, the L2 norm based loss and the log-sum-exp based loss, as showcased in Chapter 2. The relevant theoretical researches are expected. Furthermore, a more effective and comprehensive mechanism that helps to transform embeddings with respect to the loss functions is desirable.

Bibliography

[1] "Computer vision problems in plant phenotyping (cvppp2020)." https://www.plant-phenotyping.org/CVPPP2020. Accessed: 2020-06-10.

[2] C. Costa, U. Schurr, F. Loreto, P. Menesatti, and S. Carpentier, "Plant phenotyping research trends, a science mapping approach," *Frontiers in Plant Science*, vol. 9, p. 1933, 2019.

[3] D. Houle, D. R. Govindaraju, and S. Omholt, "Phenomics: the next challenge," *Nature reviews genetics*, vol. 11, no. 12, pp. 855–866, 2010.

[4] H. Scharr, M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J.-M. Pape, G. Polder, D. Vukadinovic, *et al.*, "Leaf segmentation in plant phenotyping: a collation study," *Machine vision and applications*, vol. 27, no. 4, pp. 585–606, 2016.

[5] "History of research on arabidopsis thaliana." https://en.wikipedia.org/wiki/History_of_research_on_Arabidopsis_thaliana. Accessed: 2020-06-10.

[6] A. M. Jones, J. Chory, J. L. Dangl, M. Estelle, S. E. Jacobsen, E. M. Meyerowitz, M. Nordborg, and D. Weigel, "The impact of arabidopsis on human health: diversifying our portfolio," *Cell*, vol. 133, no. 6, pp. 939–943, 2008.

[7] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter, "Annotated high-throughput microscopy image sets for validation.," *Nature methods*, vol. 9, no. 7, pp. 637–637, 2012.

[8] K. N. Niforou, A. K. Anagnostopoulos, K. Vougas, C. Kittas, V. G. Gorgoulis, and G. T. Tsangaris, "The proteome profile of the human osteosarcoma u2os cell line," *Cancer Genomics-Proteomics*, vol. 5, no. 1, pp. 63–77, 2008.

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] L. Chen, M. Strauch, and D. Merhof, "Instance segmentation of biomedical images with an object-aware embedding learned with local constraints," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 451–459, Springer, 2019.

[11] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," *arXiv preprint arXiv:1708.02551*, 2017.

[12] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8837–8845, 2019.

[13] C. Payer, D. Štern, T. Neff, H. Bischof, and M. Urschler, "Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 3–11, Springer, 2018.

[14] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Annual International Conference on Machine Learning*, pp. 41–48, 2009.

[15] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell detection with star-convex polygons," in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, pp. 265–273, 2018.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[17] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand, "Basnet: Boundary-aware salient object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7479–7489, 2019.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

[19] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, pp. 483–499, Springer, 2016.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[22] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, *et al.*, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955*, 2020.

[23] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[24] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation for autonomous driving," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[25] K. I. Appel and W. Haken, *Every planar map is four colorable*, vol. 98. American Mathematical Soc., 1989.

[26] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," *arXiv preprint arXiv:2002.10857*, 2020.

[27] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[28] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[29] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[30] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on information theory*, vol. 21, no. 1, pp. 32–40, 1975.

[31] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[32] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[35] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[36] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172, Springer, 2013.

[37] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, pp. 1–51, 2015.

[38] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, mar 2017.

[39] S. Wolf, C. Pape, A. Bailoni, N. Rahaman, A. Kreshuk, U. Kothe, and F. Hamprecht, "The mutex watershed: efficient, parameter-free image partitioning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 546–562, 2018.

[40] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," *Mathematical morphology in image processing*, vol. 34, pp. 433–481, 1993.

[41] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[42] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[43] H. Chen, X. Qi, L. Yu, and P.-A. Heng, "Dcan: deep contour-aware networks for accurate gland segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2487–2496, 2016.

[44] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5221–5229, 2017.

[45] S. Peng, W. Jiang, H. Pi, H. Bao, and X. Zhou, "Deep snake for real-time instance segmentation," *arXiv preprint arXiv:2001.01629*, 2020.

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[47] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9627–9636, 2019.

[48] C. Lim, "Mask r-cnn." https://www.slideshare.net/windmdk/mask-rcnn, 2017.

[49] W. Abdulla, “Mask r-cnn for object detection and instance segmentation on keras and tensorflow.” https://github.com/matterport/Mask_RCNN, 2017.

[50] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, “Mask scoring r-cnn,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[51] X. Zhang, G. An, and Y. Liu, “Mask r-cnn with feature pyramid attention for instance segmentation,” in *2018 14th IEEE International Conference on Signal Processing (ICSP)*, pp. 1194–1197, 2018.

[52] J. Liu and P. Li, “A mask r-cnn model with improved region proposal network for medical ultrasound image,” in *International Conference on Intelligent Computing*, pp. 26–33, Springer, 2018.

[53] A. Kirillov, Y. Wu, K. He, and R. Girshick, “Pointrend: Image segmentation as rendering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[54] Y. Lee and J. Park, “Centermask: Real-time anchor-free instance segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[55] Y. Lee, J. Hwang, S. Lee, Y. Bae, and J. Park, “An energy and gpu-computation efficient backbone network for real-time object detection,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 752–760, 2019.

[56] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, “BlendMask: Top-down meets bottom-up for instance segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[57] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and P. Luo, “Polarmask: Single shot instance segmentation with polar representation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[58] R. Zhang, Z. Tian, C. Shen, M. You, and Y. Yan, “Mask encoding for single shot instance segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[59] H. Ying, Z. Huang, S. Liu, T. Shao, and K. Zhou, “Embedmask: Embedding coupling for one-stage instance segmentation,” 2019.

[60] J.-M. Pape and C. Klukas, “3-d histogram-based segmentation and leaf detection for rosette plants,” in *European Conference on Computer Vision*, pp. 61–74, Springer, 2014.

[61] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[62] X. Yin, X. Liu, J. Chen, and D. M. Kramer, "Multi-leaf tracking from fluorescence plant videos," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 408–412, IEEE, 2014.

[63] H. Barrow, J. Tenenbaum, R. Bolles, and H. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proceedings: Image Understanding Workshop*, pp. 21–27, Science Applications, Inc Arlington, VA, 1977.

[64] S. Beucher *et al.*, "The watershed transformation applied to image segmentation," *Scanning microscopy-supplement-*, pp. 299–299, 1992.

[65] D. Ward, P. Moghadam, and N. Hudson, "Deep leaf segmentation using synthetic data," *arXiv preprint arXiv:1807.10931*, 2018.

[66] D. Ward and P. Moghadam, "Scalable learning for bridging the species gap in image-based plant phenotyping," *arXiv preprint arXiv:2003.10757*, 2020.

[67] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[68] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[69] "2018 data science bowl: Evaluation." https://www.kaggle.com/c/data-science-bowl-2018/overview/evaluation. Accessed: 2020-06-10.

[70] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi, "Semi-convolutional operators for instance segmentation," in *European Conference on Computer Vision*, 2018.

[71] V. Ulman, M. Maška, K. E. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, *et al.*, "An objective comparison of cell-tracking algorithms," *Nature methods*, vol. 14, no. 12, p. 1141, 2017.

[72] E. David, S. Madec, P. Sadeghi-Tehran, H. Aasen, B. Zheng, S. Liu, N. Kirchgessner, G. Ishikawa, K. Nagasawa, M. A. Badhon, C. Pozniak, B. de Solan, A. Hund, S. C. Chapman, F. Baret, I. Stavness, and W. Guo, "Global wheat head detection (gwhd) dataset: a large and diverse dataset of high resolution rgb labelled images to develop and benchmark wheat head detection methods," 2020.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[74] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[75] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[76] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ch. 5. MIT Press, 2016. `http://www.deeplearningbook.org`.

[77] Y. Wu, L. Chen, and D. Merhof, "Improving pixel embedding learning through intermediate distance regression supervision for instance segmentation." Manuscript submitted for publication, 2020.

[78] M. Ren and R. S. Zemel, "End-to-end instance segmentation with recurrent attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6656–6664, 2017.

[79] P. Jund, A. Eitel, N. Abdo, and W. Burgard, "Optimization beyond the convolution: Generalizing spatial relations with end-to-end metric learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, IEEE, 2018.

[80] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

A Network Architectures

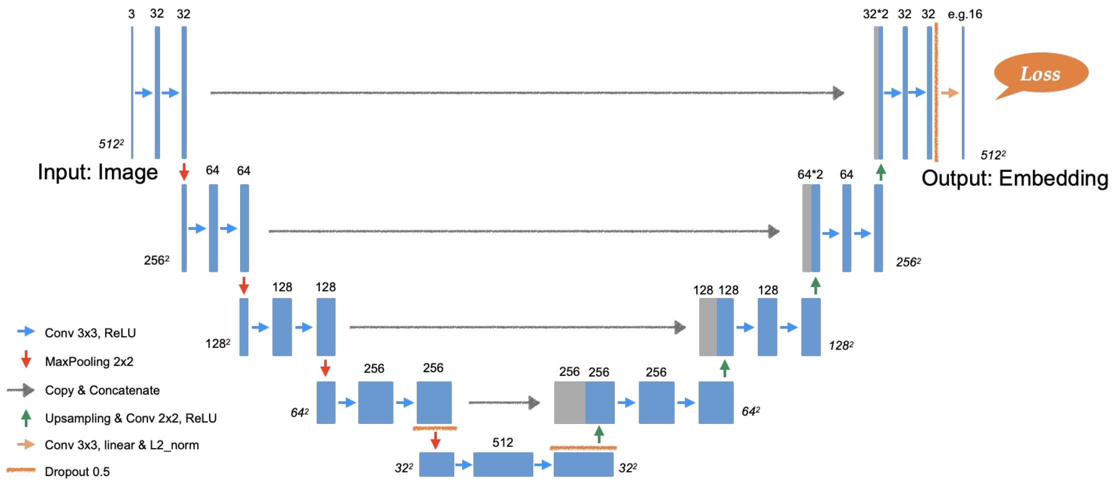


Figure A.1: Original U-Net. 2.1 revisited.

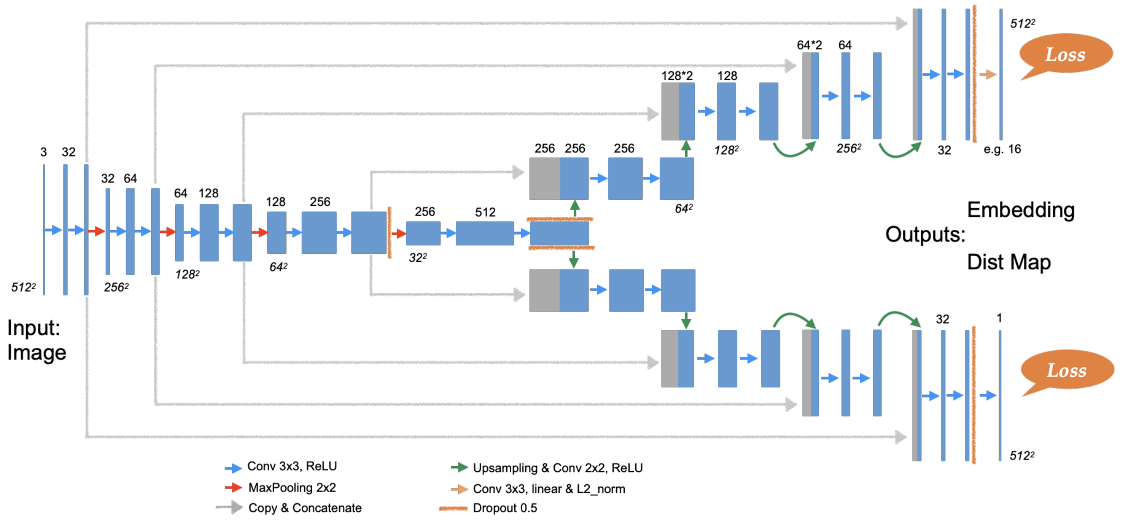


Figure A.2: U-Net with 2 Heads.

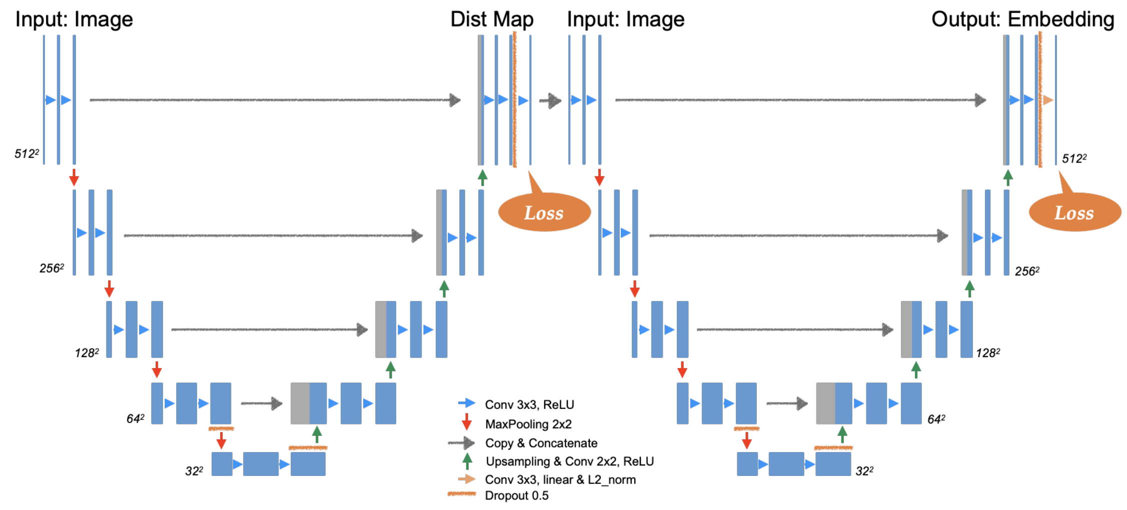


Figure A.3: Stacked U-Net (W-Net).

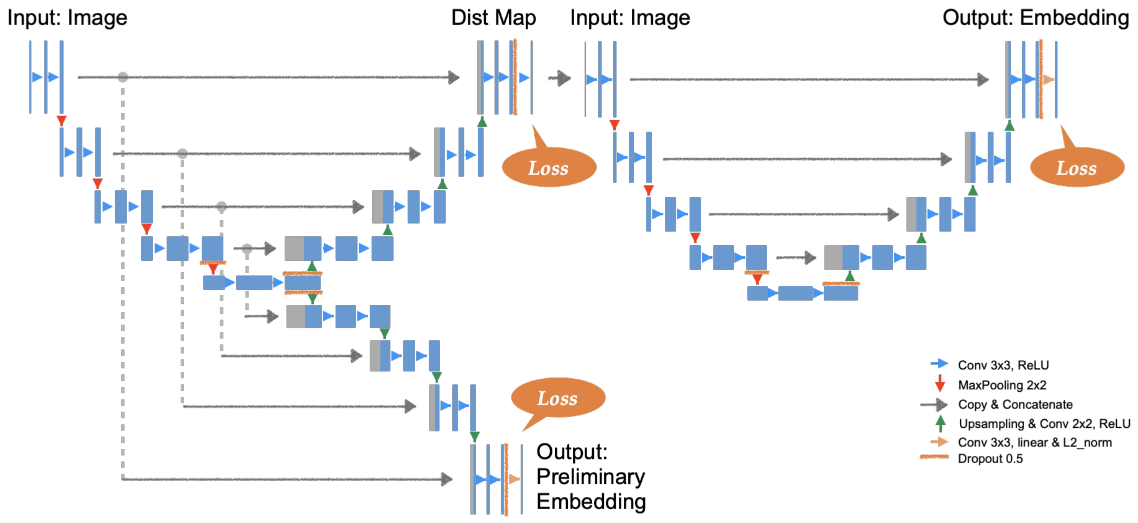


Figure A.4: Stacked U-Net (W-Net) with Intermediate Supervision.

B Extension: Ensemble Trick

During the experiments, it is found that the testing set A3 (Tobacco) is hard to obtain decent results. The most significant reason is the extremely imbalanced number of training images for Arabidopsis and Tobacco, as discussed in Chapter 5. Here, the training images of A3 are augmented and they are trained separately to investigate the performance.

Augmentation

The 27 training images are augmented to 250 images with following rules:

- horizontally mirror (probability=0.5);
- rotate (maximal ± 10 deg, probability=0.8);
- zoom (maximal $\times 4$, probability=1.0).

Ensemble

Empirically, the augmented data do not bring in much better performance: they are either improved less than 1% or even worse. However, it is found that for different trainings with random augmentation, the results are very unstable for different testing images. And roughly speaking, the larger the segmented instances are, the better the final mSBD score is. The ensemble trick is thus applied to 5 randomly chosen predictions of testing set A3, and the prediction which has the closer foreground areas to the ground truth foreground is selected.

After this ensemble trick, the mSBD from A3 can be improved from 77% to 81% and the overall mSBD can be improved from 88% to 89%. Figure B.1 illustrates some examples of with and without this ensemble trick. It is believed that this trick is not generally applicable, therefore it is considered only as an extension to the main work.

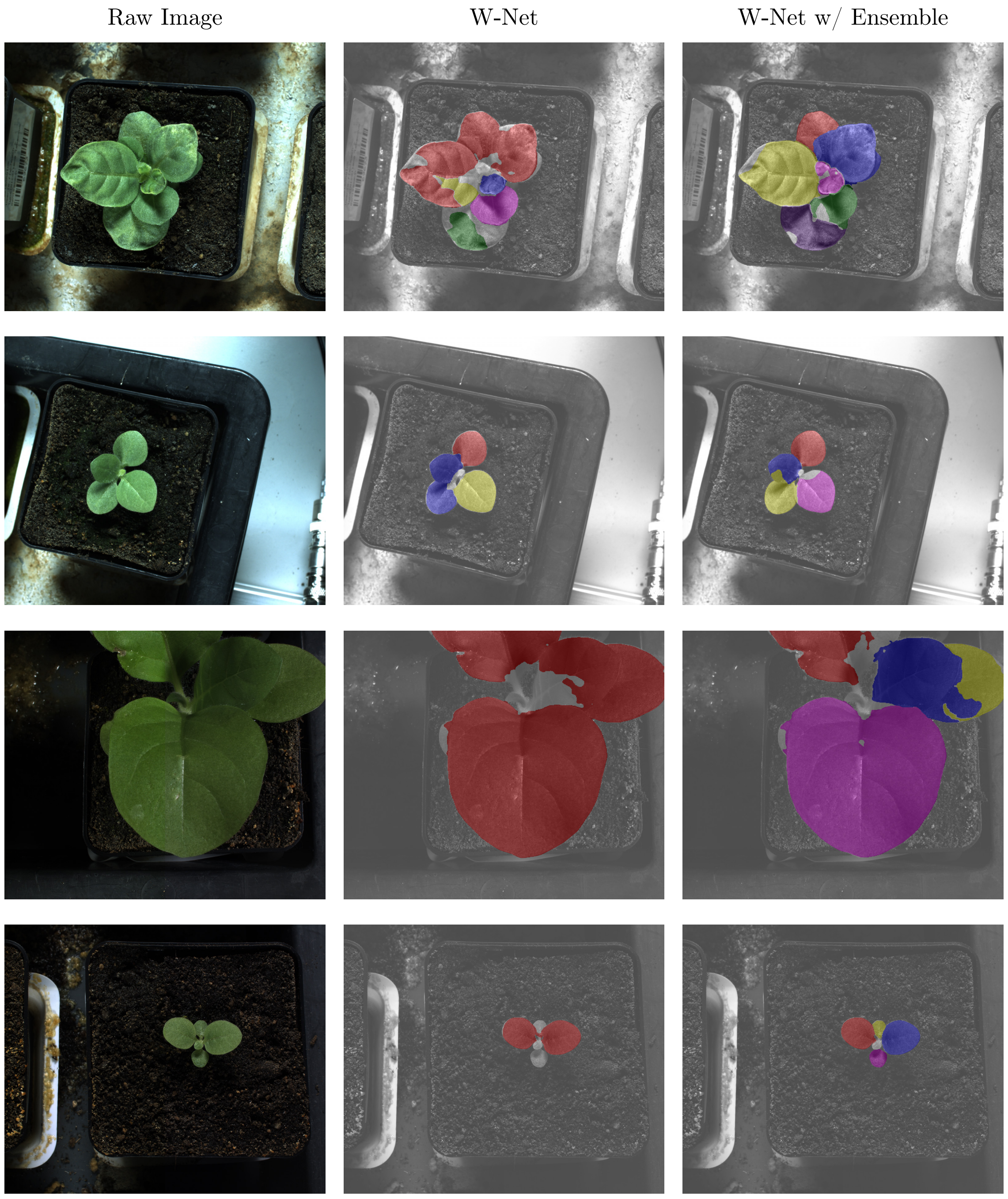


Figure B.1: Leaf Segmentation Results of the CVPPP2017 Testing Set A3: Raw Images (left), W-Net (middle) and W-Net with Ensemble (right); Each row illustrates one example. The mSBD score for testing set A3 has increased from 0.77 to 0.81.

C Online Access

Materials which are related to this thesis are permanently archived at `https://yuliwu.github.io/ma`, including:

- This thesis;
- Slides for interim presentation;
- Slides for final presentation;
- Manuscript [77] (under review) for CVPPP'20, an ECCV'20 workshop;
- Other supplemental materials.