UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ARTHUR ZACHOW

Panoptic Segmentation Using Semantic Segmentation Networks and a Discriminative Loss Function

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Cláudio R. Jung

Porto Alegre
May 2022

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos André Bulhões Mendes
Vice-Reitora: Profª. Patricia Helena Lucas Pranke
Pró-Reitora de Graduação: Profª. Cíntia Inês Boll
Diretora do Instituto de Informática: Profª. Carla Maria Dal Sasso Freitas
Coordenador do Curso de Ciência de Computação: Prof. Rodrigo Machado
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Quero dedicar este trabalho em especial à minha mãe Rosani e meus irmãos Nicolas e Natálias. Somente Deus e nós sabemos todo o esforço investido para chegarmos aqui.

AGRADECIMENTOS

First of all, I would like to thank my family for all the support during all the hardships we have faced since I started my graduation. Definitively, I would not be able to chase my dreams if were not for them. A special thank you to my uncle Martim and my grandfather Herbert, who have passed on during the pandemic, with them I had many of my most precious memories.

I would also like to thank all the professors from UFRGS, specially my advisor. Professor Claudio has given very insightful and handy feedback for this work and for my career. Even when things did not go according to the plan, he always was very patient and kind. I am very grateful for all the things I learned during the development of this work.

My friends supported me at most difficult situations I found myself in. I can not thank enough all my friends for all their help. Without their collaboration, I would not be able to get this far. I want to specially thank the Amigos Online, where all my friends helped me to stay ~~in~~sane during the pandemic.

Last but not least important. I have to thank professor Tomoko and Vinicius from the Japanese language department. Were not for them, I do not think I would have realized many of my dreams during the graduation.

ABSTRACT

Deep learning (DL) dominates the modern approaches to image segmentation. Historically, image segmentation has split into instance and semantic segmentation. When using DL-based solutions, their difference lies in the output created. The most used approaches in instance segmentation are proposal-based. They commonly use bounding boxes to represent different objects. The proposals have problems like overlapping boxes and inaccurate representation of the object's shape. Semantic segmentation generates a class identification for each pixel of the output and instance segmentation generates a proposal or an identification for the pixel. More recently, there was the definition of the task of panoptic segmentation and a metric for panoptic quality. Panoptic segmentation is similar to semantic segmentation because it generates a class identification and instance identification for every pixel in the output. In this work, we evaluate a discriminative loss function that teaches the model to cluster different instances of objects by generating an E-dimensional embedding for each pixel, and then, on inference, it executes a post-processing step that identifies these clusters. This approach has the benefits of treating the network as a black box. We analyze the results on single and multi-class datasets using the panoptic quality metric. We also explore the impacts of adding a scSE attention module on the decoder of the model on the performance of the technique.

Keywords: Panoptic Segmentation. Computer Vision. Discriminative Loss Function. Neural Networks.

Segmentação Panóptica Usando Redes de Segmentação Semântica e Uma Função de Perda Discriminativa

RESUMO

Deep learning (DL) domina as abordagens modernas de segmentação de imagens. Historicamente, a segmentação de imagens se dividiu em segmentação de instâncias e semântica. Ao se utilizar abordagens de DL, a diferença entre eles está na saída criada. As abordagens mais utilizadas para segmentação de instâncias são baseadas no uso de *proposals*. Elas, comumente, usam *bounding boxes* para representar objetos. *Proposals* têm problemas como a sobreposição das caixas e a representação imprecisa da forma dos objetos. Segmentação semântica gera uma identificação de classe para cada pixel da saída e a de instâncias, gera um *proposal* ou uma identificação por pixel. Mais recentemente, foi definida a tarefa de segmentação panóptica juntamente com a métrica que calcula a qualidade panóptica. Segmentação panóptica é similar à semântica, para cada pixel da saída são gerados identificadores de classe e instância. Neste trabalho, avaliamos uma função de perda discriminativa que ensina ao modelo a agrupar as instâncias dos objetos através da geração de uma incorporação E-dimensional para cada pixel, e então, durante a inferência, executar uma etapa de pós-processamento para identificar essas agrupações. Essa abordagem tem os benefícios de tratar o modelo como uma caixa preta. Nós analizamos os resultados em dados de classe única e múltipla utilizando a métrica de segmentação panóptica. Exploramos também os impactos da adição de um módulo de atenção scSE na performance do método.

Palavras-chave: Segmentação Panóptica. Visão Computacional. Função de Perda Discriminativa. Redes Neurais.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AV	Autonomous Vehicles
CNN	Convolution Neural Network
cSE	Channel Squeeze and Excitation
CV	Computer Vision
DL	Deep Learning
LiDaR	Light Detection and Ranging
NN	Neural Networks
PQ	Panoptic Quality
ResNet	Residual Network
RQ	Recognition Quality
SQ	Segmentation Quality
scSE	Concurrent Spatial and Channel Squeeze and Excitation
SE	Squeeze and Excitation
sSE	Spatial Squeeze and Excitation

LIST OF SYMBOLS

$\text{IoU}(x, y)$	Intersection over Union.
C	Number of classes.
$\log$	Natural log.
$y_{o,l}$	Binary indicator if the label l is the correct classification for observation o.
$p_{o,l}$	Predicted probability of the observation o for the label l.
$\|\|FN\|\|$	Number of False Negatives.
$\|\|FP\|\|$	Number of False Positives.
$\|\|TP\|\|$	Number of True Positives.
K	Number of clusters.
N_k	Number of elements in cluster k.
μ_k	Cluster center for cluster k.
δ_d	Margin for distance.
δ_v	Margin for variance.
$[x]_+$	$\max(0, x)$.
$\|x\|$	Euclidean distance (L2 norm).
E	Dimensionality of the embedding's space.
P_l	Pixel label that unifies instance and panoptic segmentation.
L_c	Class label.
L_i	Instance label.
H	Height of an image.
W	Width of an image.
D	Dimensionality of the output of the decoder's head.
L_{var}	Variance component of the discriminative loss.
L_{dist}	Distance component of the discriminative loss.

L_{reg}	Regularization term of the discriminative loss.
L_{inst}	Instance segmentation loss/discriminative loss.
L_{sem}	Semantic segmentation loss/Cross-entropy.
α	Weight for the variance term of the discriminative loss.
β	Weight for the distance term of the discriminative loss.
γ	Weight for the regularization term of the discriminative loss.
L_{total}	Combined loss value used in training and validation.
r_x	Scaling factor for the width of an image.
r_y	Scaling factor for the height of an image.

CONTENTS

1 INTRODUCTION

With the advent of Autonomous Vehicles (AV), there is an increasing demand for more precise scene comprehension algorithms in order to achieve higher levels of driving automation, as defined by the Society of Automotive Engineers. This need arises from the complex real-life situation that these AVs find in everyday driving. The vehicle computer must correctly identify pedestrians, other traffic participants, and obstacles to offer a safe experience to all people and property involved. It must extract the information from its surroundings using a set of sensors such as cameras and LiDaR, like the one shown in Figure 1.1.

Figure 1.1: An AV Sensor

Source: DuckDuckGo Images

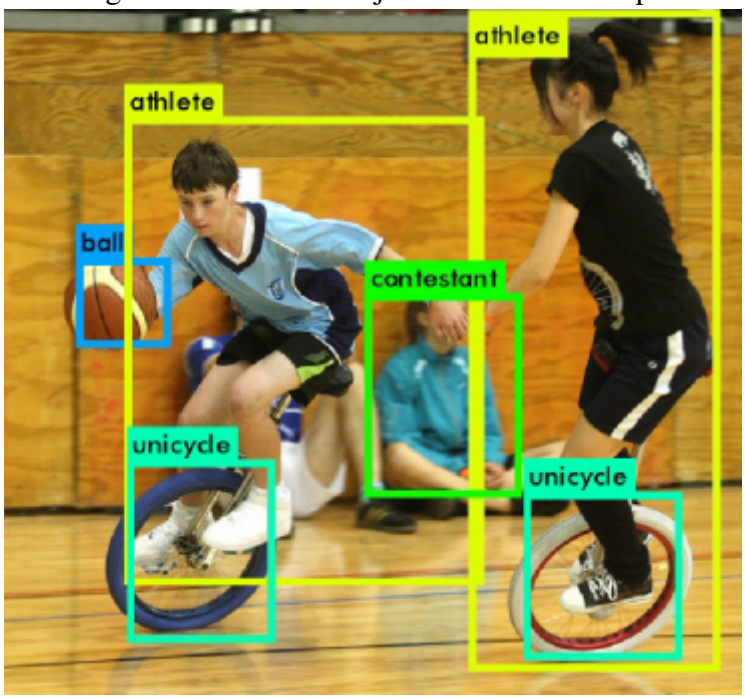
The most popular approaches nowadays for object detection aim to find a *bounding box* around each instance of a pre-defined set of categories, as illustrated in Figure 1.2. Bounding boxes provide a rough representation of the object's shape and location, but they present limitations:

- There is bounding box overlapping which causes pixel sharing by multiple objects. This problem presents a challenge for tasks like measuring these techniques with human annotators and the learning itself.
- The bounding box might contain a large area of the background instead of the object itself, particularly for articulated objects.

These problems ultimately mean that the shape of objects cannot be accurately defined

when using proposal-based solutions such as bounding boxes.

Figure 1.2: YoloV2 Object Detection Example



Source: Redmon and Farhadi (2017)

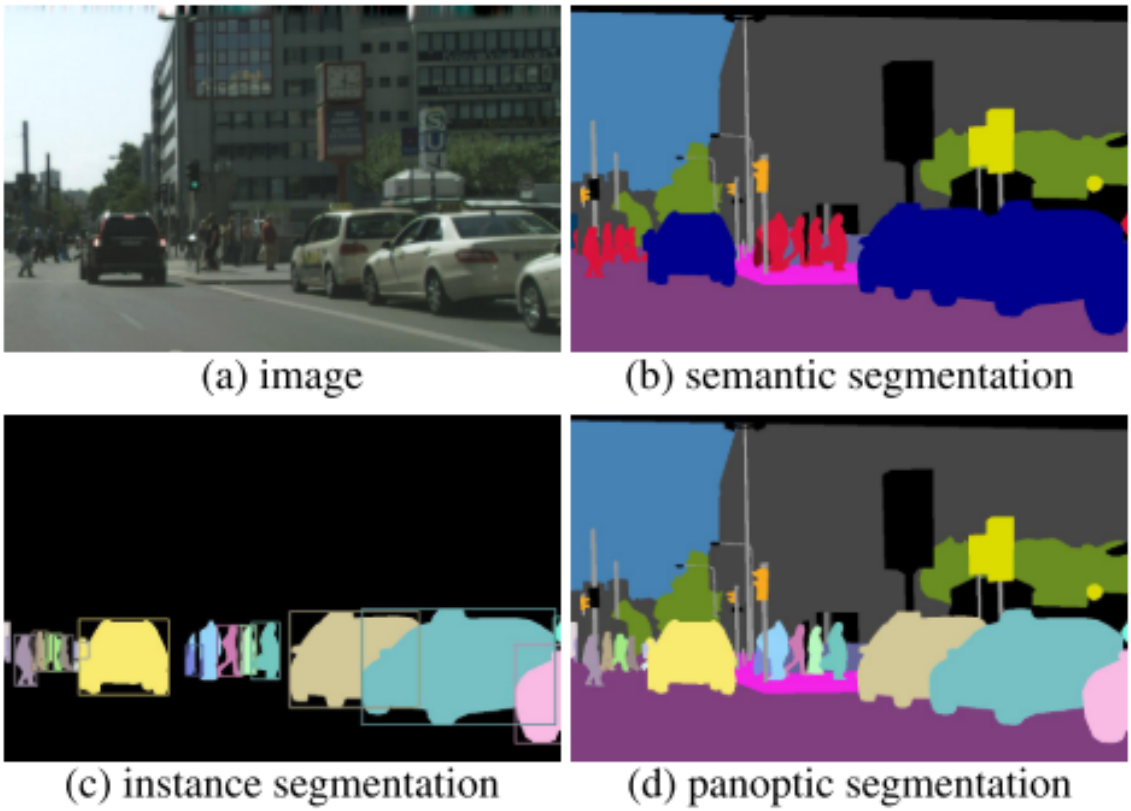
By using segmentation algorithms, we can extract information regarding the objects and composition of the scene. In general terms, segmentation allows us to segment distinct parts of the image from the rest. For example, it enables us to distinguish between persons and the road by assigning different labels to person- and road-related pixels.

Figure 1.3 illustrates different segmentation results from the same source image. As shown in Figures 1.3(b)-(d), there are different ways to define what segmentation means, which typically relates to the goal application in which the segmentation task must be applied. As Kirillov et al. (2019) explain, existing segmentation approaches are based on *things* or *stuff*. *Things* are countable objects (e.g. people or cars), and existing algorithms aim to detect each object and delineate it with a segmentation mask (instance segmentation). *Stuff* is comprised of amorphous regions of similar texture or material (e.g. sky or grass), and existing algorithms aim to assign a class label to each pixel in an image (semantic segmentation). Kirillov et al. (2019) propose a new task that aims to unify the two tasks into a new one, called *panoptic segmentation*, that presents a holistic approach to the problem.

For semantic segmentation, the problem consists of assigning a unique class-related label to each image pixel, considering a pre-defined set of categories. On the other hand, instance segmentation must handle a variable and an unknown number of in-

stances, and existing approaches typically regress an embedding feature vector (KONG; FOWLKES, 2017) that is similar for pixels belonging to the same instance, but different for those related to different instances.

Figure 1.3: Example of Different Segmentations



Source: Kirillov et al. (2019)

De Brabandere, Neven and Van Gool (2017) propose a new loss function that teaches the neural network to cluster different instances of objects in the scene based on the semantic prediction. This approach treats the neural network as a black box, where there is no need to change anything besides adding a new head to the network to output the embeddings consumed by the loss function.

In this work, we implement the technique proposed by De Brabandere, Neven and Van Gool (2017) and evaluate the effect of changing some parameters. These changes include the following:

- Usage of the complete CVPPP dataset.
- Usage of a pre-trained encoder on the larger ImageNet (RUSSAKOVSKY et al., 2015) dataset.
- Introduction of a Spatial and Channel Squeeze and Excitation (scSE) (ROY; NAVAB;

WACHINGER, 2019) attention module to the decoder blocks of the model.

The results are measured and analyzed using a panoptic segmentation approach defined by Kirillov et al. (2019).

There are two objectives for this work. First objective is to evaluate the approach proposed by De Brabandere, Neven and Van Gool (2017) through a panoptic perspective. Secondarily, we use the scSE on the model in an exploratory study to analyze how it performs in a multi-task network.

This work follows the following structure: Chapter 2 discusses the relevant technical background and the related works. Chapter 3 introduces our proposed solution. The experimental setup and the evaluation of our results are presented in Chapter 4. In Chapter 5, we present our conclusions.

2 BACKGROUND AND RELATED WORK

In this chapter, we introduce the necessary background information and some of the relevant papers on image segmentation. We will also present existing datasets for instance and panoptic segmentation.

2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN)-based methods have been widely explored for several computer vision tasks, and in particular image segmentation. The CNN architecture is still very successful and popular to this day, and many of the top-performing techniques in popular datasets such as COCO are CNN-based.

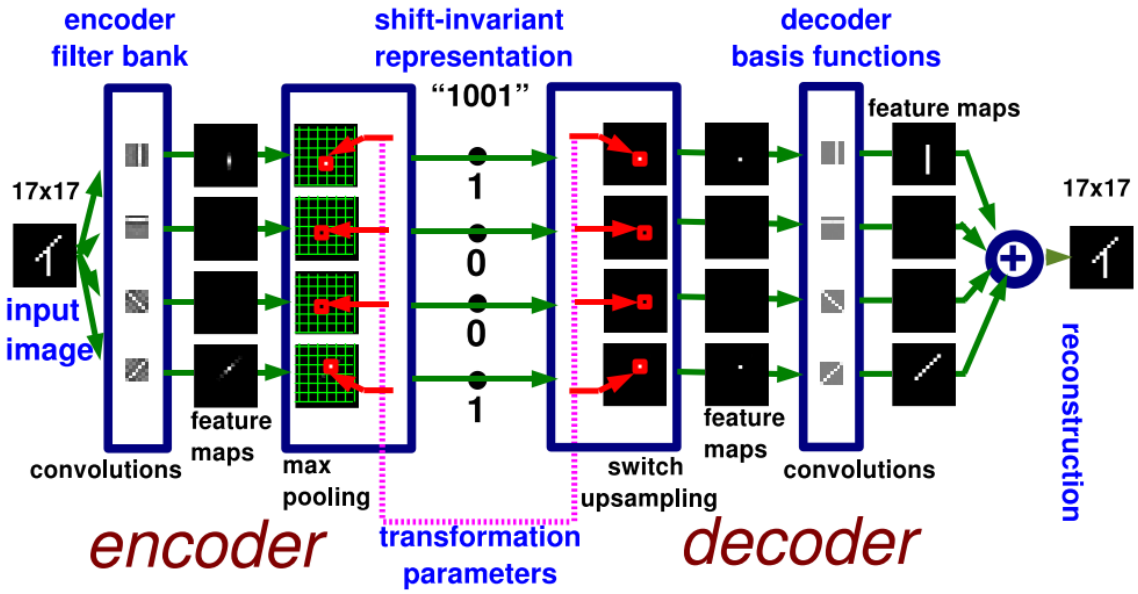
A CNN is a neural network (NN) that employs convolutions in the place of matrix multiplication in at least one layer, and are commonly used as building blocks for encoder-decoder structures. We discuss the encoder-decoder structures relevant to this work in the following subsection.

2.1.1 Encoder-decoders

Ranzato et al. (2007) introduced the first encoder-decoder network. This network is an unsupervised method for learning a hierarchy of feature detectors invariant to distortions and shifts. Their work stems from the need for robust learning from a limited amount of labeled data. This robustness of existing techniques using other methods could not handle that variability to learn invariant representations of features. There were supervised learning approaches at the time that could learn invariant representations, but they suffered from hyper parametrization.

Ranzato et al. (2007) proposed a hierarchical method in which each layer was composed of a convolution filter and a downsampling pooling layer (known today as max pooling). This layer may be hierarchically stacked to learn ever deeper invariant features without supervision. Figure 2.1 shows the architecture of a encoder-decoder. The encoder first convolutes the input and then follows with a max-pooling operation. This stage creates the invariant representation. The decoder then uses the representation (the what) and the transformation parameters (the where) to produce a reconstruction.

Figure 2.1: Encoder-Decoder Architecture



Source: Ranzato et al. (2007)

We focus on the ResNet (HE et al., 2016) encoder and the decoder of the U-Net (RONNEBERGER; FISCHER; BROX, 2015) because we use them in this work.

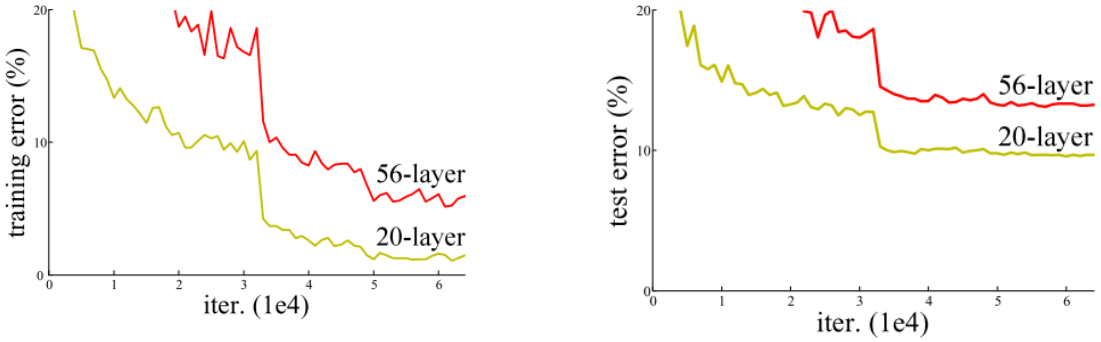
2.1.1.1 ResNet Encoder

The encoder, also known as the backbone or feature extractor, is the part of the network responsible for creating "abstractions" on top of the input data. The intuition behind the usage of encoders is that they transform the variable-sized input data into a fixed size state. When implementing neural networks, the networks can extract more abstract data by stacking these encoders into deeper networks.

Increasing the depth of the encoder yields higher-level feature extraction, but choosing the ideal depth of the encoder is not trivial. The expectation is that increasing the encoder depth would result in better performance. In reality, this happens up to a certain depth before worsening the model accuracy. He et al. (2016) claimed that the increase in depth can lead to vanishing gradients when training the network, as illustrated in Figure 2.2. More precisely, this figure shows the training and test errors for an object recognition task varying the depth of the network, and increasing the depth also increased the error. To mitigate this problem, He et al. (2016) proposed a Residual Network (ResNet) model by using skip connections to feed the gradients from the last layers to the first ones.

The ResNet architecture combines the first layer with four other blocks. The first layer contains a convolution with batch normalization and max-pooling. The rest are

Figure 2.2: Effect of the Vanishing Gradients



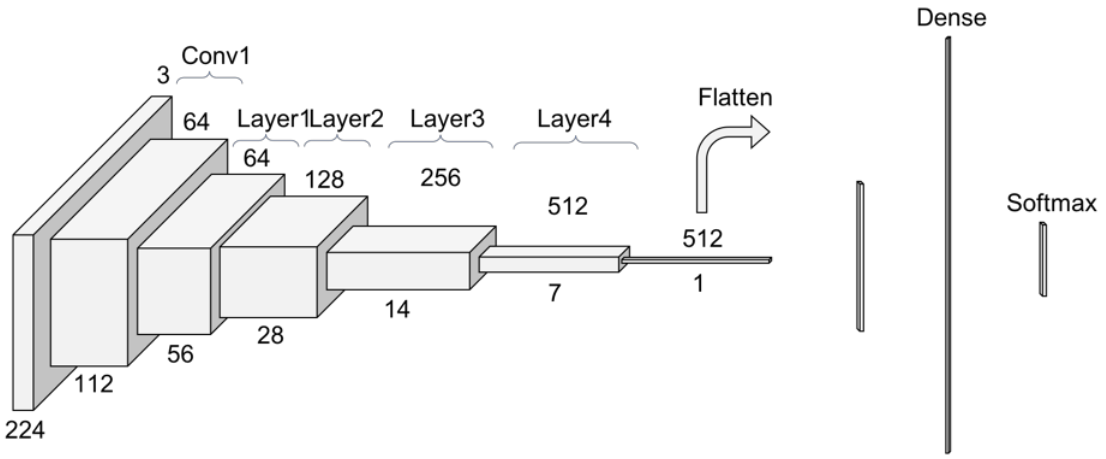
(a) Training error of the trained network.

(b) Test error of the same network.

Source: He et al. (2016)

composed of either basic or bottleneck blocks. The number of operations in each block differentiates these two types. An operation consists of convolution, batch normalization, and a ReLU activation. The only exception is the last block which does not have the ReLU activation. Figure 2.3 showcases the overall architecture of the encoder. The number of layers identifies the variations of the ResNet.

Figure 2.3: The ResNet Encoder Architecture



Source: Pablo Ruiz (2018)

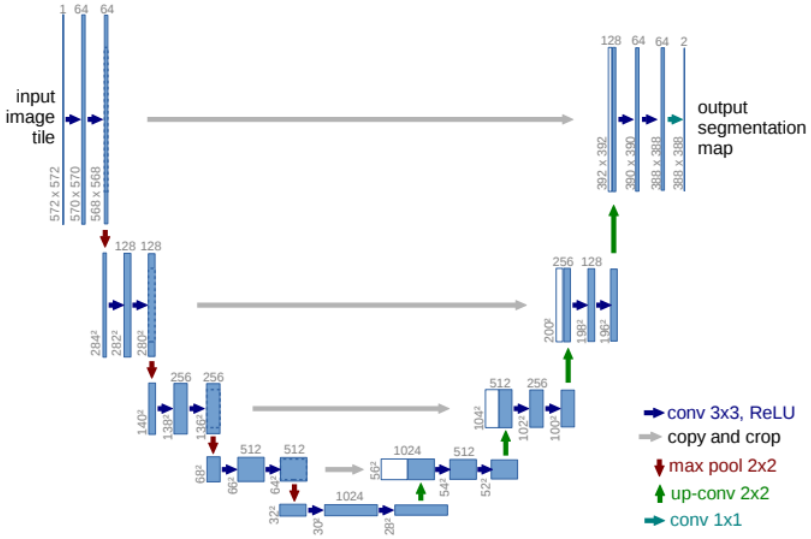
2.1.1.2 Decoder

The decoder is the part of the network that can translate the abstract state created by the encoder into something "usable", such as the reconstruction of the input image. In our case, that is not yet the network's output: it is a D-dimensional feature map that feeds the two different heads of the network for semantic and instance segmentation. The specific architecture of our models is described in Section 3.1.

Ronneberger, Fischer and Brox (2015) created the U-Net to get a class identification for each pixel of the input image (semantic segmentation). Traditionally, the classification task was able to identify what was on the input image, but with poor localization. Their results indicate that using the method proposed improved training times and required a lot fewer input images to achieve satisfactory results.

U-Nets mitigate this problem by combining the high-resolution features from the encoder path with the upsampled output. The decoder effectively mirrors the encoder, and upsampling transforms the outputs to the desired size. The upsampling increases the spatial dimensions with the drawback of losing context due to the feature space's reduction. Figure 2.4 showcases the baseline decoder block. Convolution, batch normalization, and ReLU make up the basic decoder block. The result of the block is either upsampled to increase resolution or passed forward to the heads of the network.

Figure 2.4: The U-Net Architecture



Source: Ronneberger, Fischer and Brox (2015)

2.1.2 Attention

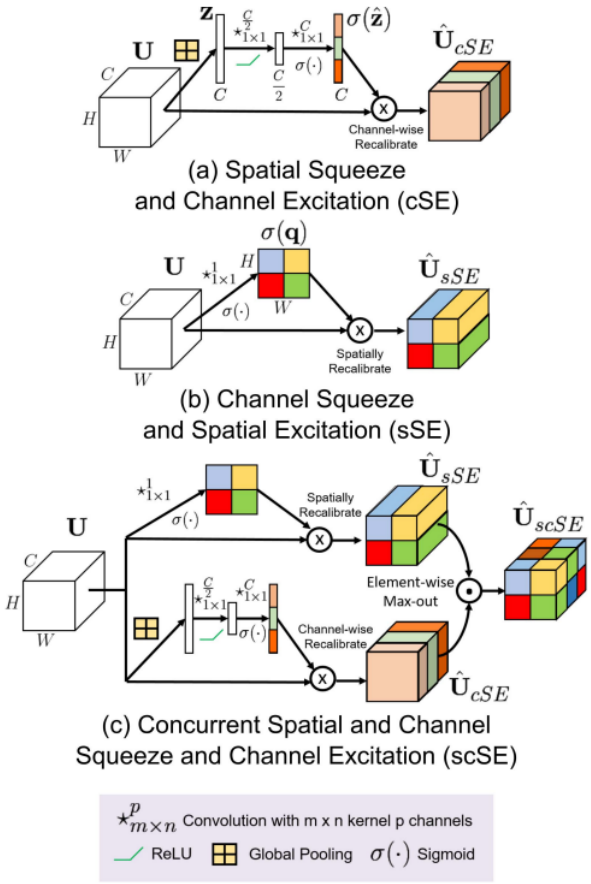
Attention modules in neural networks work similarly to the human brain. They try to weigh more relevant things and diminish the lesser ones by recalibrating the feature maps. A segmentation model can use these modules on the encoder, decoder, or both.

There are many different types of attention, differing by their object of attention. Guo et al. (2022) presents a survey discussing many of the existing attention types. In this work, we chose to use scSE attention because according to Roy, Navab and Wachinger

(2019) it adds little complexity to the model with significant performance improvements in dense semantic segmentation tasks. The main objective of the work developed by Roy, Navab and Wachinger (2019) was to create a complementary module to the Squeeze-and-Excite (SE) layer (HU et al., 2017) that improved performance in dense segmentation tasks. It works on the channels domain and is spatially independent.

Essentially, the scSE module creates masks in the spatial and channel domains separately, and then combines them. It stems from two other independent modules, one of channel-wise SE (cSE) and the other of spatial SE (sSE), both of them proposed by Roy, Navab and Wachinger (2019). The method tries to maximize the flow of information into the attention module. Figure 2.5 shows the architecture of these modules.

Figure 2.5: Architecture of the SE Modules



Source: Roy, Navab and Wachinger (2019)

2.2 Datasets

As in any supervised learning approach, annotated data is required to train the models. In this work, we have used two datasets that support panoptic segmentation. These datasets have pixel-wise annotation for semantic class and instance identifiers (id). We note that Kirillov et al. (2019) mention in their work that datasets that support panoptic segmentation have existed even before the formal definition of the task.

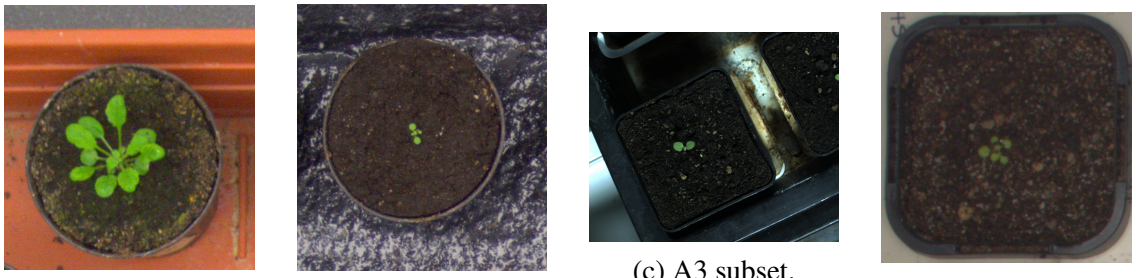
We use one of the datasets recommended by Kirillov et al. (2019) and another that is considerably smaller and simpler. The goal is to validate our assumptions and implementation on the smaller dataset and then scale the approach on the bigger one.

2.2.1 CVPPP Leaf Segmentation Challenge 2017

Minervini et al. (2016) presented a dataset of densely annotated images for plant phenotyping: CVPPP Leaf Segmentation Challenge 2017. This dataset aims to encourage the usage of computer vision approaches to segment each leaf of the plants. Four subsets, each with their setup, compose the dataset, totaling 810 samples. This dataset provides supplementary data for all leaves about their center, quantity, bounding box, and an instance mask. It also has semantic information for all the leaves and the input image for every sample.

Figure 2.6 shows one sample for each subset of this dataset. Notice the differences between the setup and the plants. The leaves and the background are the only semantic classes in this dataset. It fits our experiments perfectly because it provides a challenging but relatively simple testbed to validate our assumptions and implementation. Because the training data is scarce, we supplement it with data augmentation.

Figure 2.6: Samples of the CVPPP Dataset



(a) A1 subset. (b) A2 subset. (c) A3 subset. (d) A4 subset.

Source: The Authors

2.2.2 Cityscapes

Cordts et al. (2016) created a dataset for image segmentation in urban contexts: Cityscapes. It consists of many different subsets, and the subset of interest for our work is the finely annotated images. This subset consists of 5,000 images, of which 2,975 are training images, 500 are validation images, and 1,525 are test images. The dataset supplements all samples with i) instance segmentation information, ii) semantic masks, and iii) bounding boxes. There are 19 different classes represented in this dataset, including *things* (i.e., people or cars) and *stuff* (i.e., buildings or sidewalks). The annotations combine category-level information with instance-level information through a single label

$$P_l = \begin{cases} 1000L_c + L_i, & \text{if } L_c \text{ is a } \textit{thing}, \\ L_c, & \text{otherwise} \end{cases}, \tag{2.1}$$

where L_c is the label of the class and L_i is the instance label. Note that *stuff* objects do not present individual instances, and *things* objects might present up to 1,000 instances for any given class. We use this format again to generate images based on the model predictions implemented in this work.

This dataset is very challenging for the methods tested in this work. It exposes the models to near real-world scenarios and fits them perfectly within the context of this work.

2.3 Traditional Image Segmentation

According to Szeliski (2011), the task of segmentation consists of finding groups of pixels that go together. It is one of the oldest problems in computer vision, and the work of Brice and Fennema (1970) is one of the earliest works in automated image segmentation. Many applications use image segmentation nowadays, like medical imaging, Autonomous Vehicles (AVs), and defense systems.

There are many different categories of algorithms developed for image segmentation in the literature. Some of these are based on simple thresholding (OTSU, 1979), region growing (NOCK; NIELSEN, 2004), active contours (KASS; WITKIN; TERZOPOULOS, 1988), and k-means clustering (DHANACHANDRA; MANGLEM; CHANU, 2015). They aim to perform a partition of the input image into regions that present similar visual

information such as color and texture, but typically semantic information is not explored and the results might not produce object instances, since the same object might contain different parts with variable visual information.

Clustering groups similar data points together to identify groups or regions of coherent data. In particular, mean-shift clustering (COMANICIU; MEER, 2002) is an interesting approach since it does not require a previous knowledge on the number of clusters, and will be briefly described next.

2.3.1 Mean Shift

Mean shift is essentially a clustering algorithm that aims to group together "similar" feature vectors, but can be applied to image segmentation by considering both color and positional features (COMANICIU; MEER, 2002). Figure 2.7 shows the application of the Mean Shift algorithm to segment the input image.

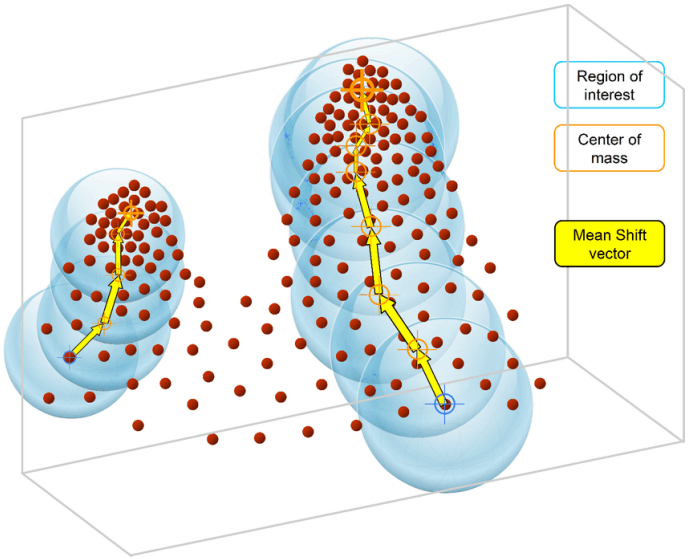
Figure 2.7: Image Segmentation with Mean Shift Clustering

Source: <https://github.com/Rasoul77/MeanShiftImageSeg>

The Mean Shift algorithm introduced by Comaniciu and Meer (2002) was based on earlier work (FUKUNAGA; HOSTETLER, 1975). The most important aspect of it is that it is a non-parametric algorithm that does not require any knowledge besides the data itself to use. Mean Shift is a centroid-based algorithm that updates the centroid and slowly reaches the regions with high data density related to local modes of the distribution. The algorithm has a parameter called bandwidth, which is the region used to update candidates for the centroid. In Figure 2.8, there is a visual description of the algorithm, were the center of mass is the centroid and the region of interest (spheres) is the bandwidth. According to Comaniciu and Meer (2002), because this algorithm does not assume

spherical clusters, it is better suited for real-world problems because the data derived from the real world have arbitrary shapes.

Figure 2.8: Visualization of the Mean Shift Clustering Algorithm



Source: Chen et al. (2018)

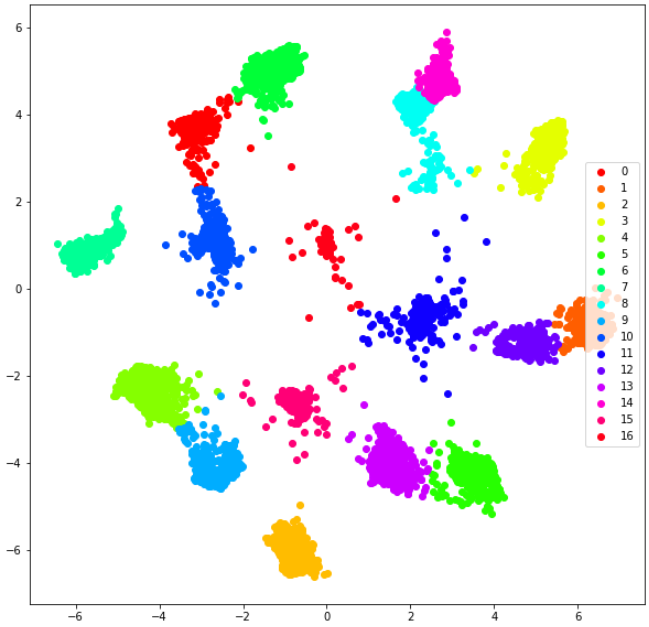
The Mean Shift algorithm does not require prior knowledge of the number of clusters, but the distribution of the data in feature space has an enormous impact on the convergence time for this algorithm. Having well-separated clusters is important, and the use of a suitable loss function for the problem of instance segmentation can provide cluster separability. We discuss this specific topic of the relation between the discriminative loss and the clustering performance in more detail in Section 3.2.2. In Figure 2.9, we show an example of the application of the Mean Shift algorithm for 2D instance-related embeddings, which are the output of the method proposed by De Brabandere, Neven and Van Gool (2017). Each entry in the legend table on the right represents the color of each identified cluster.

2.4 More Modern Views on Image Segmentation

Image segmentation is not very well defined, particularly because identifying objects in an image is subjective. For example, when you see a forest, do you need to identify each leaf, tree or the whole forest as one object? In their work, Kirillov et al. (2019) loosely defines the two existing segmentation tasks and introduces a new one that unifies the other two.

The historical division between semantic segmentation and object detection stems

Figure 2.9: Example of the Embeddings in 2D Space



Source: The authors

from the early history of computer vision (CV). Kirillov et al. (2019) defines the objects of study of each segmentation task as being *stuff* and *things*, respectively. The authors define *stuff* as amorphous regions of similar texture or material. Examples of *stuff* are the sky, grass, or roads. Because these regions are uncountable by nature, the task of studying them focuses on pixel-wise class identification. This task assigns every pixel a class label.

Things, on the other hand, constitutes the domain of countable objects. For example, these objects are people or cars. Object detection with precise boundaries is also known as instance segmentation. It focuses on separating instances of the different classes with bounding boxes or segmentation masks. The result of this task might be instance-wise using the bounding boxes approach or pixel-wise assigning some pixels a class and instance label.

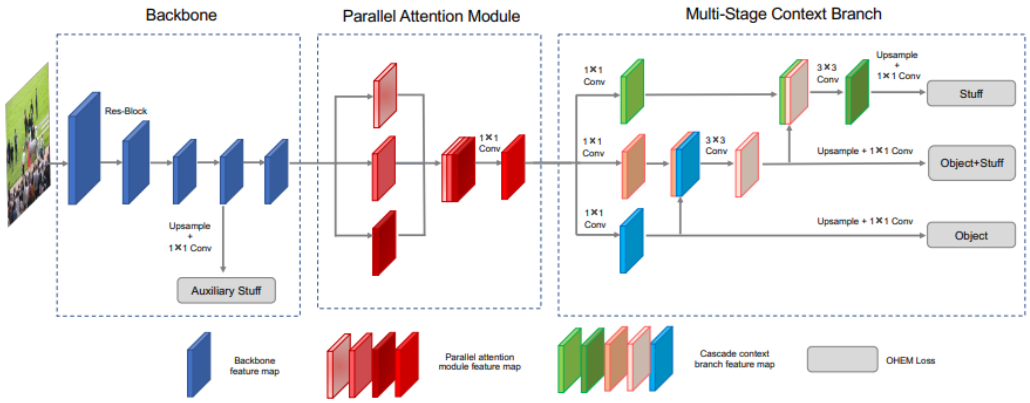
Although the tasks seem very similar, they are very different. Their datasets, details, and evaluation metrics are almost totally unrelated. With that in mind, Kirillov et al. (2019) defines the panoptic segmentation task. It aims to merge semantic and instance segmentation into one that holistically tackles image segmentation. This task assigns for every pixel a class and instance label.

2.4.1 Existing Panoptic Segmentation Techniques

Currently, the state of the art in panoptic segmentation is hybrid/multi-task approaches (WANG et al., 2019; CHEN et al., 2019). The multi-task approach predates the definition of the panoptic task. It aims to combine the results of the instance segmentation with the class segmentation task to produce a panoptic segmentation of the image. Kirillov et al. (2019) cautions that although the final objective is the same, the multi-task approach is not panoptic because it is not a unified view of the problem. It may also allow inconsistencies between *things* and *stuff*. The approach studied in this work is also a multi-task approach to the panoptic segmentation problem.

The work of Wang et al. (2019) is currently the top performer in the COCO dataset panoptic segmentation leaderboard. Their method is based on the usage of a CNN with an encoder, a parallel attention module, and a multi-stage context branch. Their architecture is shown in Figure 2.10.

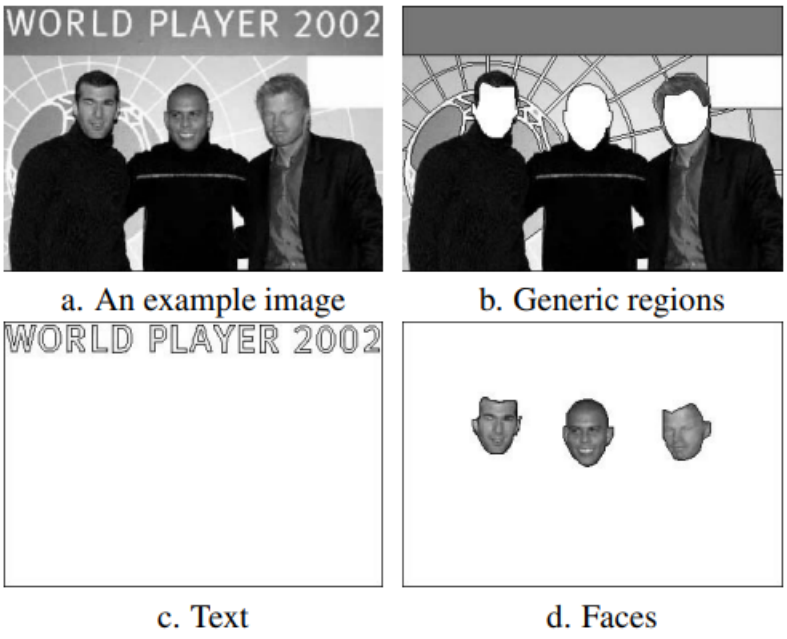
Figure 2.10: Megvii's Architecture



Source: Wang et al. (2019)

There are also approaches from before the rise of deep learning approaches. These approaches suffered largely from the lack of definition for the panoptic task. One of such examples is the seminal work of Tu et al. (2003) where they introduce the Image Parsing method to extract faces and text from images. Their approach proposes a general Bayesian framework for joint segmentation, detection, and recognition. Image Parsing is panoptic segmentation because it does not split the task into different subtasks such as instance or semantic segmentation. Figure 2.11 shows the application of the method.

Figure 2.11: Example of Face and Text Detection



Source: Tu et al. (2003)

2.4.2 Panoptic Quality Metric

Kirillov et al. (2019) defined the panoptic segmentation task and a suitable metric for assessment. This metric is called the Panoptic Quality (PQ) and has two stages. It offers a unified and interpretable view of the panoptic segmentation task. The existing metrics were not suited for this task because they cannot evaluate semantic and instance segmentation. PQ is also insensitive to class imbalance.

The first stage is called segment matching. It provides a unique matching between ground truth and prediction segments. This matching defines three different sets: the true positives (TP), false negatives (FN), and false positives (FP). Figure 2.12 demonstrates the definition of the three sets.

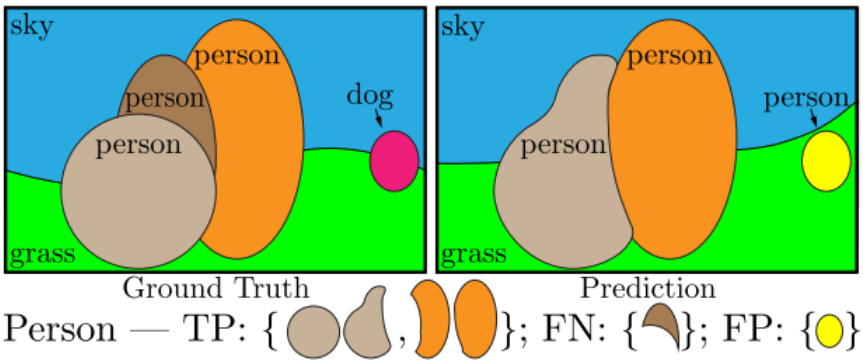
With the three sets calculated from the segment matching, we can compute the PQ through:

$$PQ = \frac{\sum_{(p,g)\in \mathrm{TP}} \mathrm{IoU}(p,g)}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}. \tag{2.2}$$

One important characteristic about PQ is that it aims to be easily interpretable. It can be re-written as segmentation quality (SQ) and recognition quality (RQ)

$$PQ = \underbrace{\frac{\sum_{(p,g)\in \mathrm{TP}} \mathrm{IoU}(p,g)}{|\mathrm{TP}|}}_{\mathrm{SQ}} \times \underbrace{\frac{|\mathrm{TP}|}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}}_{\mathrm{RQ}}, \tag{2.3}$$

Figure 2.12: Segment Matching Example for Person Class



Source: Kirillov et al. (2019)

and we can see the performance of the model for both things and stuff. The alternate view also allows us to evaluate specific aspects of the segmentations more pertinent to one or other heads of the model. Similar to the model's output, the SQ and RQ are not independent of each other because SQ is measured only over the matched segments.

3 THE PROPOSED METHODOLOGY

In our work, we aim to reproduce the work of De Brabandere, Neven and Van Gool (2017) to analyze their results using the PQ metric and to evaluate the impact of adding an attention module to our network's decoder.

We implemented their discriminative loss function using the same parameters as the baseline work on a U-Net with ResNet34 backbone. We also study the impact of two changes: i) the addition of scSE attention modules to the network decoder aims to explore global information, and ii) the variation of the dimensionality of the instance embedding vectors. The PQ metric evaluates the results quantitatively.

We chose the methodology proposed by De Brabandere, Neven and Van Gool (2017) because it is a simple approach that produces satisfying results. The simplicity is that it builds on top of any semantic segmentation model, treating it as a black box. Because De Brabandere, Neven and Van Gool (2017) have proposed the technique before the definition of the panoptic segmentation task by Kirillov et al. (2019), we reproduce and evaluate their experiments using the PQ metric.

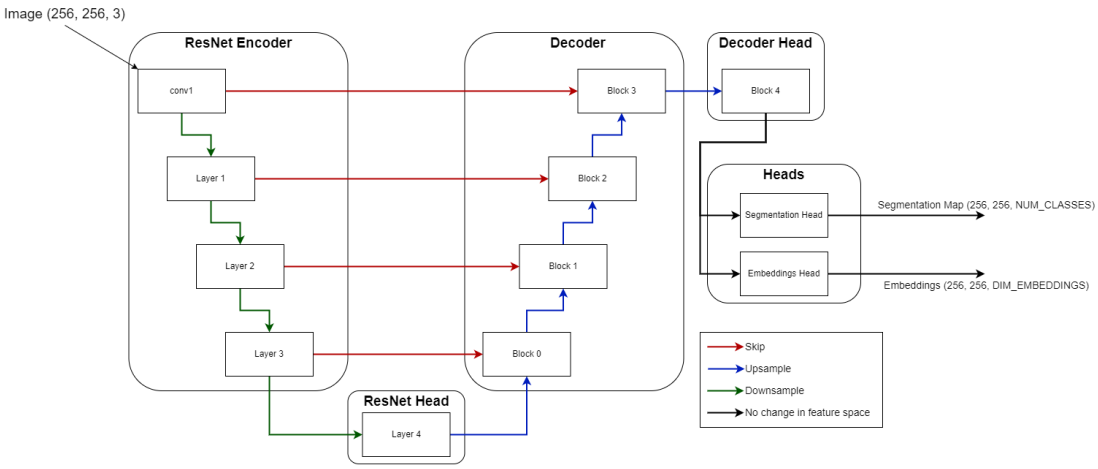
The proposed modifications present two main goals: i) theoretically, it should be visible in the components of the PQ the dependency of the multi-task approach because the instance segmentation uses as an input the semantic segmentation results; ii) attention layers can give the model a global perspective of the features when decoding, which might improve the model capacity to differentiate between similar objects.

3.1 Model Structure

The model structure follows the baseline (De Brabandere; NEVEN; Van Gool, 2017) with a modification to the decoder part of the architecture, and the baseline implementation uses the U-Net architecture. It uses ResNet34 encoders, typical U-Net decoders, and presents two heads at the end. One of these heads is responsible for the semantic segmentation part, and the other for the instance segmentation, as illustrated in Figure 3.1. The dimensions of the feature maps are 64, 128, 256, 512, 1024. This is ordered in the order they appear in the encoder, so the input of Block 1 is 64. We added the attention modules after each block of the decoder.

The network requires two predictions per pixel to achieve panoptic segmentation. The segmentation head will produce a class label, while the embeddings head will create

Figure 3.1: Baseline Architecture Implemented



Source: The Authors

embeddings that we use to derive the instance label. They are built on top of the decoder path and consume the D-dimensional feature map it produces. Both heads have a similar architecture with the only difference in the number of channels of the output. They are composed of a 1×1 convolution layer. The activation function is different for each head: the semantic uses a softmax activation and the instance head uses a linear activation.

The segmentation head uses convolution operations to translate the input feature map into the class-map prediction for all pixels. This head produces an output tensor with dimensions $C \times H \times W$, where C is the number of categories.

However, it must produce an E-dimensional embedding vector for each pixel (with linear activation) so that pixels related to the same instance should present similar embedding vectors, and pixels related to distinct ones should present distant embedding vectors. It produces a $E \times H \times W$ output tensor.

3.2 Loss Functions

Because we approach the panoptic segmentation task with a multi-task learning technique, we must have a separate loss function to handle the corresponding task. We next explain our choices for the loss functions.

3.2.1 Semantic Segmentation Loss

The semantic segmentation problem reduces to a classical classification problem. Network designers often choose the cross-entropy function as a baseline for this problem. We choose to use this function because we need a good baseline for the semantic segmentation part of our multi-task learning. Jadon (2020) explains in the survey that loss functions that perform well in many different datasets are good baselines to use when the data distribution is unknown. The cross-entropy loss makes one of such baselines even though it assumes a perfect balance of classes.

The cross-entropy measures the difference between two probability functions given a variable or set of events. This function is practical when using a dataset with an arbitrary quantity of classes. Its formula is given by:

$$L_{\text{sem}} = -\sum_{c=1}^{C} y_{o,l} \log\left(p_{o,l}\right), \tag{3.1}$$

where: L_{sem} is the semantic loss value. $y_{o,l}$ is a binary indicator if the label l is the correct classification for observation o. C is the number of categories/classes. $\log(o, l)$ is the natural log.

3.2.2 Instance Segmentation Loss

De Brabandere, Neven and Van Gool (2017) proposed in their work a new loss function based on other methods that used *discriminative* losses to optimize the distance between different images. The authors have created a new loss function that optimizes the distances between distinct objects of the same class in a picture. The objective of this function was to be a simple method of having instance segmentation on top of semantic segmentation networks.

The core idea of the discriminative loss is to produce clusters of embeddings that are part of the same objects. In this manner, a clustering method can use its result to create instance segmentation. The loss presents two parameters: the margin for variance denoted as δ_v, and the margin for distance δ_d. The margin for variance penalizes sparse embeddings, where the embeddings of one instance are too far from the other embeddings of that same instance. The margin for distance penalizes when the clusters are too close to each other. Effectively, this loss function aims to create densely populated regions in

the embedding space related to each cluster. Each of these dense regions (clusters) ideally represents an instance of an object.

Three terms compose the loss function: the variation measures the clusters uniformity, the distance measures the clusters separation, and the regularization penalize arbitrarily big clusters, they are given by:

$$L_{\text{var}} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{i=1}^{N_k}[||\mu_k - x_i|| - \delta_v]_+^2, \tag{3.2}$$

$$L_{\text{dist}} = \frac{1}{K(K-1)}\sum_{\substack{k_a=1 \\ K_a!=K_b}}^{K}\sum_{k_b=1}^{K}[2\delta_d - ||\mu_{k_a} - \mu_{k_b}||]_+^2, \tag{3.3}$$

$$L_{\text{reg}} = \frac{1}{K}\sum_{k=1}^{K}||\mu_k||, \tag{3.4}$$

and the combined discriminative loss is given by

$$L_{\text{inst}} = \alpha L_{\text{var}} + \beta L_{\text{dist}} + \gamma L_{\text{reg}}, \tag{3.5}$$

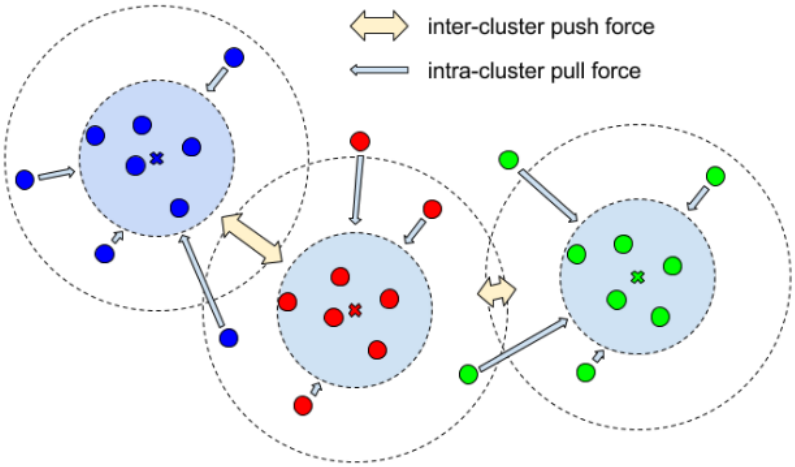
Where: L_{inst} is the instance segmentation loss. K is the number of clusters. N_k is the number of elements in cluster k. μ_k is the centroid of the cluster. x_i is an embedding. δ_v is the margin for variance. $[x]_+$ is equal to $\max(0, x)$. δ_d is the margin for distance. L_{var} is the term for variance of the clusters. L_{dist} is the term for the distance between the clusters. L_{reg} is the term of regularization. α, β and γ are weights for each of the terms.

The parameters δ_v and δ_d are not independent. Their relation can be seen in Figure 3.2 where the inter-cluster push force is parametrized by δ_d and the intra-cluster pull is parametrized by δ_v.

In the work of De Brabandere, Neven and Van Gool (2017), the authors demonstrate that if the variance and distance terms are zero, and if the following relation is true $\delta_d > 2\delta_v$, then all embeddings are closer to their instances than they are to any other. This condition theoretically guarantees that this loss is capable of instance segmentation. The regularization term aims to keep clusters centroids near the origin of the embedding space.

The weights used in α, β and γ do not significantly impact the method's performance. Accordin to De Brabandere, Neven and Van Gool (2017), their values are usually 1, 1 and 0.001, respectively.

Figure 3.2: Cluster Forces



Source: De Brabandere, Neven and Van Gool (2017)

3.2.3 Total Loss

Both loss functions need to be combined to generate a total loss function for the network. Both losses have the same weight on the total loss. The following formula defines the total loss function L_{total} for the training and validation stages:

$$L_{\text{total}} = L_{\text{sem}} + L_{\text{inst}} \tag{3.6}$$

Where: L_{total} Is the combined loss of both tasks.

3.3 Post-Processing

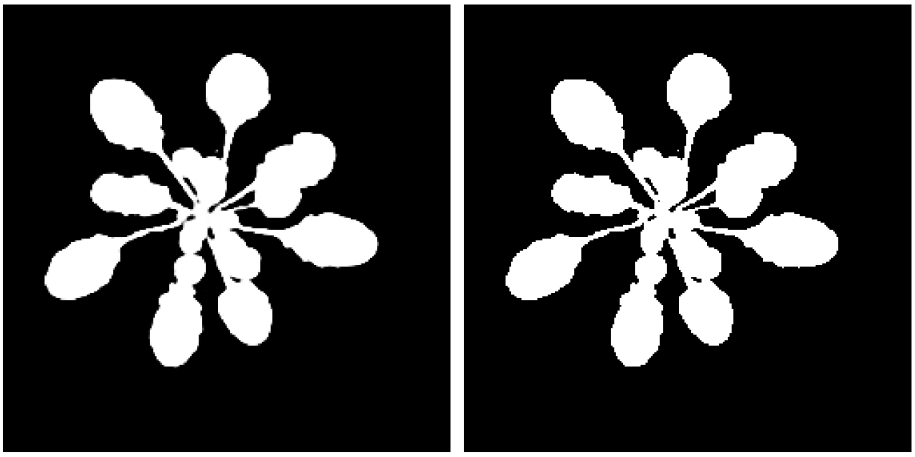
To achieve panoptic segmentation, we need to generate a class id and an instance id for every pixel. When retrieving a prediction from a given sample, the model generates for us two different outputs. We, therefore, need to combine these two. Our approach is roughly equal to the one described by De Brabandere, Neven and Van Gool (2017), detailed next.

We first create a class mask for each sample. In this stage, we must avoid class overlapping. We consider that a pixel's class is the highest prediction score. From the results of the semantic segmentation head, we retrieve the class with highest prediction score for every pixel.

Figure 3.3 demonstrates the effects of the class masking step for a two-category

problem. Note how the edges of the segmentation become jagged. The white color represents the leaves, and black represents the background. The aliased effect is due to the fuzzy borders of the leaves becoming either leaves or background.

Figure 3.3: Class Masking Result



(a) The leaf channel of the prediction. (b) Class mask of the prediction.

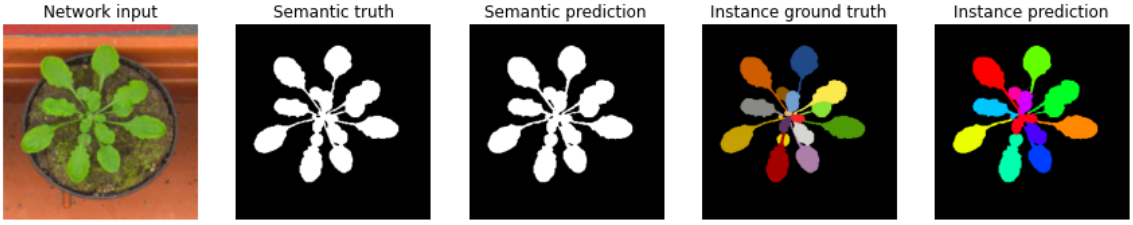
Source: The Authors

The second step consists of clustering the embeddings for instance segmentation using Mean Shift, as done in De Brabandere, Neven and Van Gool (2017). The motivation for this specific algorithm is that it requires no previous knowledge about the number of clusters (instances) in the input, making it a good candidate for real-world applications where this information is not known beforehand.

The class mask allows us to cluster all pixels of the same category and identify different instances based on the embedding vectors. The bandwidth parameter for the mean shift clustering is the δ_d used by the discriminative loss function, so that ideally it should only contain samples from the same cluster.

Figure 3.4 shows an example of the result of the clustering. Note that the instance coloring in the ground truth and the prediction are not related and are arbitrary.

Figure 3.4: Mean Shift Clustering on Model's Output



Source: The Authors

3.4 Evaluation

The PQ metric evaluates the trained models' performance. We have used only the publicly available data from the CVPPP and Cityscapes datasets to analyze the models. The evaluation did not use the testing sets because they did not have annotations.

We implement data augmentation on the datasets to avoid overfitting and improve model generalization. We split the data into the different sets before the data augmentation step because this order guarantees that the samples selected to replicate are not from the testing samples. The operation happens in this specific order to avoid data leaking.

The augmentation increases the number of training samples on the CVPPP Leaf Segmentation Challenge 2017 dataset. Even though the Cityscapes dataset has enough entries to train the network, we still use it to improve the robustness of the model with variations of the training samples.

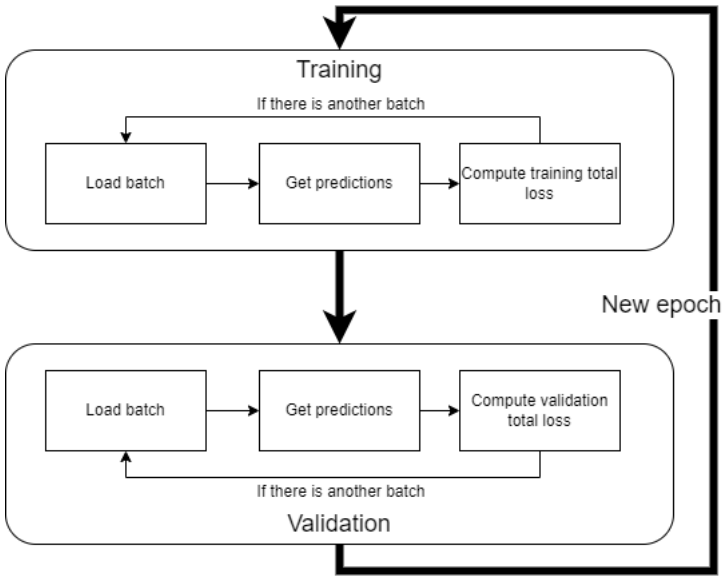
4 EXPERIMENTAL RESULTS

In this chapter, we present the experimental setup and the pipelines of training and evaluation. We describe the technologies used in the experimentation and the specific aspects of the data preparation. The experimental results are at the end of the chapter.

4.1 Training

We used Google Colab to train and evaluate all the models. Therefore, the service constrained the experiments performed. As a consequence of that, the execution hardware of the experiments changed multiple times. The experiments required the usage of GPUs for all stages involved in the model. In Figure 4.1, we show graphically the training sequence.

Figure 4.1: Training Loop



Source: The Authors

The setup of all experiments was similar, except for the dataset preparation step. We explain in the following sections the differences in the handling and their motivations.

4.1.1 Dataset Preparation and Augmentation

All datasets required pre-processing before usage in the training or other stage of the experiments. This stage included relabeling the data and splitting it into different sets.

The first stage was relabeling. This stage involved removing the unused labels for training on the Cityscapes dataset. The dataset owners provide the set of scripts used in this stage. We prepare the masks for instances and classes to support panoptic segmentation in this stage.

Panoptic segmentation requires the tuple (L_c, L_i) for every pixel. The class masks provide a class label L_c that is an integer c in the range $\{0, 1, ..., C-1\}$, where C is the number of classes, and the instance mask provides a label L_i is an integer in the range $\{0, 1, ..., I_c - 1\}$, where I_c is the number of instances belonging to category c in the training image.

There are multiple instance maps for each sample, one for each class present. An instance map is a $H \times W$ array with one integer value representing which instance is at that pixel. This instance map derives from the ground truth information. Each semantic mask is a $H \times W$ array with one integer value representing the class identification. Due to the unlabeled data's existence in the Cityscapes dataset, there might be some gaps in the evaluated images during the evaluation stage.

The datasets split used were not provided by the owners of the datasets. For the CVPPP, we have used all available data, which are 810 image samples. We have split this dataset into 60% for the training set, and the other 40% are divided equally between validation and testing sets. We used the fine annotations Cityscapes dataset containing 3,475 images, excluding the unlabeled test data. The train split of the Cityscapes was divided again into 80% for training data and 20% for validation. The original validation set of 500 images tested our models' performance.

4.1.2 Data Augmentation

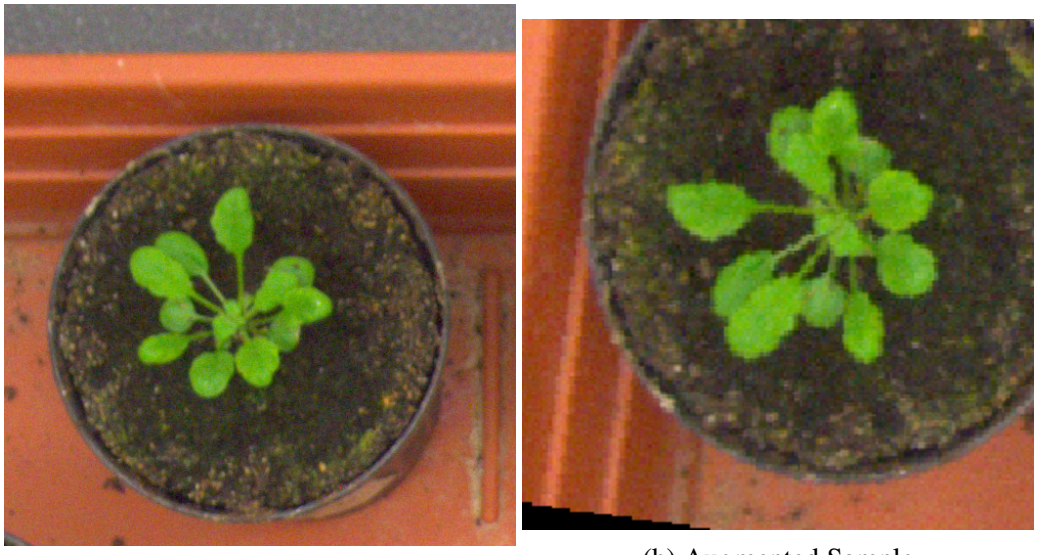
We must be aware of overfitting when training machine learning methods. This situation arises from the loss of the model generalization ability, in which the training accuracy is high but there is a significant degradation in accuracy of unseen samples. If the model has enough abstraction power and you train it almost infinitely, then it is possible to fit the training data perfectly. We use data augmentation and an early stopping strategy to mitigate this problem in our work.

Data augmentation increases the number of samples for the CVPPP Leaf Segmentation Challenge 2017 dataset because it has too few images for training. We use resizing, random rotation, random horizontal flip, and random rescaling with cropping. In the ran-

dom rescaling augmentation we have two parameters that are used to create a new image with dimensions $r_y H \times r_x W$, then we randomly choose a window to crop it such that the output image has the same dimensions as the input. We do the augmentations on the source images used only in training to avoid data leakage. Furthermore, we normalize input image so that each channel has a mean of zero and a variance of one in all subsets.

Even though the number of samples in the training set of the Cityscapes dataset is sufficiently big, we use data augmentation to improve the model robustness. Examples of the augmentation are shown in Figure 4.2.

Figure 4.2: Example of the Augmentations Used



(a) Original Input Sample (b) Augmented Sample

Source: The Authors

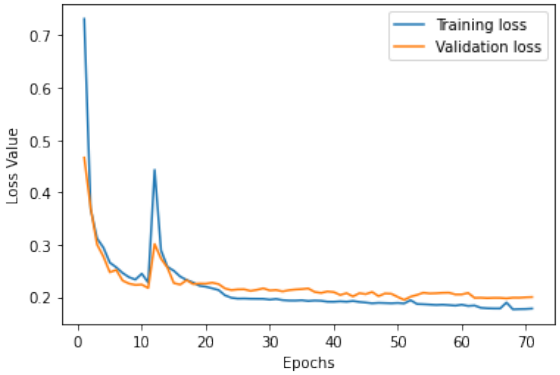
4.1.3 Early Stopping

During the training phase of the models, we have adopted an early stopping strategy. The aim is to avoid overfitting the model to the training datasets used. This strategy uses a patience approach that keeps track of the validation loss. If this validation loss does not decrease any further, the training is interrupted.

Figure 4.3 showcases a typical training loop that was interrupted by early stopping. See that the validation loss curve follow the training loss and are slightly worse overall. This graph represents a situation where there is no overfitting. If the model were overfitting the training data, the validation loss would increase while the training would lower. The spike in the training loss in the figure is bigger than the validation is due to the

augmentations performed in the training data.

Figure 4.3: Training Losses



Source: The Authors

In our experiments, we have set the patience value to 20. So if the validation loss did not decrease after 20 epochs, the training would be interrupted. In that regard, the number of training epochs was usually below 100.

4.2 Training Details

The discriminative loss has two major configurable parameters, as explained in 3.2.2. For the experiments, we use the values used by De Brabandere, Neven and Van Gool (2017) for the two datasets.

We have used Yakubovskiy (2020) implmentation as the base for our own. It allowed us to use pre-trained ResNet encoders on the Imagenet dataset in all the experiments realized. We froze the gradients of the encoder because it already could encode the inputs sufficiently well. The weights provided are sufficient because other people trained it on a much larger dataset with similar classes to those in our datasets.

In the augmentations, we have resized the images on the CVPPP dataset to 256×256 and for the Cityscapes dataset it was 256×512. The angle of rotation applied was chosen randomly in the range of $[0, 2\pi]$. The range of values for the deformation was $[1, 1.5]$.

Table 4.1 describes all the parameters used for the experiments. We selected the optimizer and learning rate based on what worked best experimentally. The values for α, β and γ were 1, 1 and 0.001 as suggested by De Brabandere, Neven and Van Gool (2017).

Table 4.1: Experiment's Fixed Parameters

δ_v	δ_d	**Learning Rate**	**Optimizer**	**Weights**	**Patience**
0.5	1.5	3e-3	AdamW	ImageNet	20

4.3 Proposed Experiments and Results

Table 4.2 summarizes the set of experiments performed in this work. Basically, the experiments follow the baseline work using or not the attention module.

Table 4.2: Baseline Experiments

Network	**Dataset**	**Embeddings Dimensionality**
U-Net	CVPPP	16
	Cityscapes	8
U-Net w/ scSE	CVPPP	16
	Cityscapes	8

In Tables 4.3 and 4.4, we present the experiments' results. The results are measured by the PQ metric and are broken down into its components. Results are shown in three different categories. First is the overall result of the evaluation considering all classes involved, the other results are specific for *things* and *stuff* following the definition by Kirillov et al. (2019).

Table 4.3: Results on the CVPPP Dataset

	All			**Things**			**Stuff**		
Model	**PQ**	**SQ**	**RQ**	**PQ**Th	**SQ**Th	**RQ**Th	**PQ**St	**SQ**St	**RQ**St
Baseline	81.4	91.6	87.9	63.6	83.9	75.8	99.3	99.3	100
scSE	83.9	92.2	90.2	68.4	85.0	80.5	99.4	99.4	100

In Table 4.3, the results show an overall improvement of around 3%. This improvement is in line with results reported by Roy, Navab and Wachinger (2019) that report a similar improvement in the task performance. Visually, on Figures 4.4 and 4.5, we can see that the usage of the scSE module improved the segmentation of the smaller leaves.

We can see that the biggest improvement with the attention module was on the RQ term. The model's capacity to recognize the different leaves was improved by the scSE module. This fact suggests that this module can recalibrate the feature maps even on multi-task architectures.

The results for the *things* (the leaves) class shows us that the segmentation of instances does not perform very well even with the semi-perfect *stuff* (the background) segmentation. The smaller or overlapping leaves are mistakenly labeled as the same instance. The reason for this might be that our architecture does not provide a sufficient

number of layers dedicated to the instance segmentation and, because of that, it does not have enough abstraction power to learn sufficiently distinct embeddings.

Figure 4.4: Results of the Inference on Some Samples for the Baseline Model

Source: The Authors

Table 4.4: Results on the Cityscapes Dataset

	All			Things			Stuff		
Model	**PQ**	**SQ**	**RQ**	**PQTh**	**SQTh**	**RQTh**	**PQSt**	**SQSt**	**RQSt**
Baseline	14.5	29.2	18.9	0.6	15.1	0.9	24.5	39.4	31.9
scSE	16.0	35.9	20.5	0.7	22.4	1.1	27.1	45.7	34.5

The results for the experiments using the Cityscapes dataset could not be performed optimally. Due to the size of the images and the considerably larger dataset, the model could not be trained sufficiently. This problem occurred due to Colab's session unreliability even when using the service on two accounts with the best subscription plans.

Figure 4.5: Results of the Inference on Some Samples for the Modified Model

Source: The Authors

The sessions would expire before the training was over as well as the service restricted access to the necessary hardware. In both Cityscapes experiments, the number of epochs that could be reliably trained was 80.

Figure 4.6 shows the results of the panoptic prediction on the COCO dataset format. The yellowish color represents the car class, which it seems to be able to distinguish but there are no instances of it segmented. If there were, it would be represented as a variation on the tonality of the yellow color used to represent the cars, this seems in accordance to the results of the PQ metric for the *things* classes.

Visually, *stuff* segmentation can somewhat delineate the overall shape of vegetation and roads. The results for the *things* are badly affect by these imprecise results of the semantic segmentation.

Overall, the resulting models for the Cityscapes dataset underperformed in comparison to the baseline work by De Brabandere, Neven and Van Gool (2017). This performance is justified by the under training of the models.

Even though the Cityscapes results are inconclusive, it is a good example of the inconsistency problems of the multi-task approach mentioned by Kirillov et al. (2019). In summary, this method of creating instance labels cannot handle the pixels that received the wrong semantic label. During the training, the model use the ground truth mask of classes and instances to train the embeddings, this way the model learns to map these activations to a certain region of embedding's space. If the model encounters a pixel that is wrongly classified during inference, it's embedding will in a completely different region of the embedding's space. The problem is even worse if the correct label of the pixel is a *stuff* class. The embeddings for these pixels are random because they are randomly initialized and are not trained at all. In the end, the clustering result will be completely different because of the "pollution" in the embedding's space.

Figure 4.6: Predictions on the Cityscapes Dataset

(a) RGB Input.

(b) Baseline prediction.

(c) scSE prediction.

Source: The Authors

5 CONCLUSION

In this work, we have analyzed the method proposed by De Brabandere, Neven and Van Gool (2017) through the PQ metric. We have implemented the multi-task approach created by De Brabandere, Neven and Van Gool (2017), which implements instance segmentation on top of the semantic segmentation.

In our experiments, We have trained a model based on the U-Net architecture that uses the ResNet38 encoder. Another variation of this model was also tested, it included the addition of scSE attention modules after each of the decoder's blocks.

The models were trained on two different datasets. The CVPPP dataset is a single class dataset that we used to validate our approach. The Cityscapes is a bigger dataset with 19 semantic classes, 8 of these being *things* classes.

The CVPPP dataset's results shows us that the usage of the attention module produces results in line with what is reported in the work of Roy, Navab and Wachinger (2019) of around 3-5% improvement in the overall PQ performance. The attention module improves the recognition of similar objects such as small leaves or overlapping leaves. This module was conceived for the semantic segmentation task but it performs well on the multi-task setting.

The results of the Cityscapes dataset are inconclusive for the usage of scSE on this dataset. The environment used for experimentation is unable to train sufficiently the models because the time to train the network is longer than the maximum allowed session length in Colab. These experiments showed a particular flaw of the discriminative loss, that is its poor handling of incorrectly labeled embeddings.

Regarding the future work, there are a few different options to look into. Below we describe a few of these options and their rationale.

The study of the dimensionality of the embeddings is an interesting option because although the different classes are treated independently in the discriminative loss function, more complex datasets might benefit from the increase in the embedding's dimensionality.

Different network models could be used in order to connect the different tasks. It would be interesting to test variations where the heads for the different tasks are serialized instead of parallel, this way the network might learn their relation to each other.

REFERENCES

BRICE, C. R.; FENNEMA, C. L. Scene analysis using regions. **Artificial Intelligence**, Elsevier, v. 1, n. 3-4, p. 205–226, jan 1970. ISSN 00043702.

CHEN, C. et al. Joint COCO and Mapillary Workshop at ICCV 2019: COCO Panoptic Segmentation Challenge Track Technical Report: Panoptic HTC with Class-Guided Fusion. 2019.

CHEN, W. et al. Airborne LiDAR remote sensing for individual tree forest inventory using trunk detection-aided mean shift clustering techniques. **Remote Sensing**, MDPI AG, v. 10, n. 7, jul 2018. ISSN 20724292.

COMANICIU, D.; MEER, P. Mean shift: A robust approach toward feature space analysis. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 24, n. 5, p. 603–619, may 2002. ISSN 01628828.

CORDTS, M. et al. The Cityscapes Dataset for Semantic Urban Scene Understanding. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, IEEE Computer Society, v. 2016-Decem, p. 3213–3223, apr 2016. ISSN 10636919. Available from Internet: <https://arxiv.org/abs/1604.01685v2>.

De Brabandere, B.; NEVEN, D.; Van Gool, L. Semantic Instance Segmentation with a Discriminative Loss Function. aug 2017. Available from Internet: <http://arxiv.org/abs/1708.02551>.

DHANACHANDRA, N.; MANGLEM, K.; CHANU, Y. J. Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm. **Procedia Computer Science**, Elsevier, v. 54, p. 764–771, jan 2015. ISSN 18770509.

FUKUNAGA, K.; HOSTETLER, L. D. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. **IEEE Transactions on Information Theory**, v. 21, n. 1, p. 32–40, 1975. ISSN 15579654.

GUO, M. H. et al. Attention mechanisms in computer vision: A survey. **Computational Visual Media**, v. 14, n. 8, 2022. ISSN 20960662. Available from Internet: <https://github.com/MenghaoGuo/Awesome-Vision-Attentions>.

HE, K. et al. Deep residual learning for image recognition. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, IEEE Computer Society, v. 2016-Decem, p. 770–778, dec 2016. ISSN 10636919. Available from Internet: <https://arxiv.org/abs/1512.03385v1>.

HU, J. et al. Squeeze-and-Excitation Networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Computer Society, v. 42, n. 8, p. 2011–2023, sep 2017. ISSN 19393539. Available from Internet: <https://arxiv.org/abs/1709.01507v4>.

JADON, S. A survey of loss functions for semantic segmentation. **2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2020**, 2020. Available from Internet: <https://github.com/shruti-jadon/>.

KASS, M.; WITKIN, A.; TERZOPOULOS, D. Snakes: Active contour models. **International Journal of Computer Vision**, Springer, v. 1, n. 4, p. 321–331, jan 1988. ISSN 09205691. Available from Internet: <https://link.springer.com/article/10.1007/BF00133570>.

KIRILLOV, A. et al. Panoptic segmentation. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 2019-June, p. 9396–9405, 2019. ISSN 10636919.

KONG, S.; FOWLKES, C. Recurrent Pixel Embedding for Instance Grouping. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, IEEE Computer Society, p. 9018–9028, dec 2017. ISSN 10636919. Available from Internet: <https://arxiv.org/abs/1712.08273v1>.

MINERVINI, M. et al. Finely-grained annotated datasets for image-based plant phenotyping. **Pattern Recognition Letters**, North-Holland, v. 81, p. 80–89, oct 2016. ISSN 01678655.

NOCK, R.; NIELSEN, F. Statistical region merging. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 26, n. 11, p. 1452–1458, nov 2004. ISSN 01628828.

OTSU, N. Threshold Selection Method From Gray-Level Histograms. **IEEE Trans Syst Man Cybern**, SMC-9, n. 1, p. 62–66, 1979. ISSN 00189472.

Pablo Ruiz. **Understanding and visualizing ResNets | by Pablo Ruiz | Towards Data Science**. 2018. Available from Internet: <https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8>.

RANZATO, M. et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, 2007. ISSN 10636919. Available from Internet: <http://www.cs.nyu.edu/~yann>.

REDMON, J.; FARHADI, A. YOLO9000: Better, faster, stronger. **Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017**, Institute of Electrical and Electronics Engineers Inc., v. 2017-Janua, p. 6517–6525, dec 2017. Available from Internet: <https://arxiv.org/abs/1612.08242v1>.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 9, p. 16591–16603, may 2015. ISSN 21693536. Available from Internet: <https://arxiv.org/abs/1505.04597v1>.

ROY, A. G.; NAVAB, N.; WACHINGER, C. Recalibrating Fully Convolutional Networks With Spatial and Channel 'Squeeze and Excitation' Blocks. **IEEE Transactions on Medical Imaging**, Institute of Electrical and Electronics Engineers Inc., v. 38, n. 2, p. 540–549, aug 2019. ISSN 1558254X. Available from Internet: <https://arxiv.org/abs/1808.08127v1>.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision**, Springer New York LLC, v. 115, n. 3, p. 211–252, dec 2015. ISSN 15731405. Available from Internet: <https://link.springer.com/article/10.1007/s11263-015-0816-y>.

SZELISKI, R. **Computer Vision**. London: Springer London, 2011. (Texts in Computer Science). ISBN 978-1-84882-934-3. Available from Internet: <http://link.springer.com/10.1007/978-1-84882-935-0>.

TU, Z. et al. Image Parsing: Unifying Segmentation, Detection, and Recognition. 2003.

WANG, S. et al. Joint COCO and Mapillary Workshop at ICCV 2019: Panoptic Segmentation Challenge Track Technical Report: Explore Context Relation for Panoptic Segmentation. 2019.

YAKUBOVSKIY, P. **Segmentation models pytorch**. [S.l.]: GitHub, 2020. <https://github.com/qubvel/segmentation_models.pytorch>.