UNIVERSITAT DE BARCELONA

FUNDAMENTALS OF DATA SCIENCE MASTER'S THESIS

Unsupervised Segmentation Using CNNs Applied to Food Analysis

Author:
Montserrat BRUFAU VIDAL
Àlex FERRER CAMPO
Markos GAVALAS

Supervisor:
Petia RADEVA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamentals of Data Science

in the

Facultat de Matemàtiques i Informàtica

July 3, 2018

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Unsupervised Segmentation Using CNNs Applied to Food Analysis

by Montserrat Brufau Vidal
Àlex Ferrer Campo
Markos Gavalas

In the recent times, there have been numerous papers on deep segmentation algorithms for vision tasks. The main challenge of these tasks is to obtain sufficient supervised pixel-level labels for the ground truth. The main goal of this project is to explore if Convolutional Neural Networks can be used for unsupervised segmentation. We follow a novel unsupervised deep architecture, capable of facing this challenge, called the W-net and we test it on food images. The main idea of this model is to concatenate two fully convolutional networks together into an autoencoder. The encoding layer produces a k-way pixelwise prediction, and both the reconstruction error of the autoencoder as well as the error from the decoder are jointly minimized during training. We search for the best architecture for this network and we compare the results for this unsupervised network with supervised results from a well-known network.

Acknowledgements

Thanks to our advisor Petia Radeva for her guidance and support through all the project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, segmentation – applied to still 2D images, video, and even 3D or volumetric data – is one of the key problems in the field of computer vision. Looking at the big picture, segmentation is one of the high-level tasks that paves the way towards complete scene understanding. Such problem has been addressed in the past using various traditional computer vision and machine learning techniques including normalized cuts (Cour, Benezit, and Shi, 2005), (Shi and Malik, 2000), Markov random field-based methods (Zhang, Brady, and Smith, 2001), mean shift (Comaniciu and Meer, 2002), hierarchical methods (P et al., 2010), and many others.

Despite the popularity of those kind of methods, the deep learning revolution has turned the tables so that many computer vision problems – segmentation among them – are being tackled using deep architectures, usually Convolutional Neural Networks (CNNs) segmentation, which are surpassing other approaches by a large margin in terms of accuracy and sometimes even efficiency. While convolutional networks have already existed for a long time (LeCun et al., 1989), their success was limited due to the size of the available training sets and the size of the considered networks. The breakthrough by Krizhevsky et al. (Krizhevsky, Sutskever, and Hinton, 2012) was due to supervised training of a large network with 8 layers and millions of parameters on the ImageNet dataset with 1 million training images. Since then, even larger and deeper networks have been trained (Simonyan and Zisserman, 2014). The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. Typically, these methods are trained using models such as fully convolutional networks to produce a pixel-wise prediction. Supervised training methods can then be employed to learn filters to produce segments on novel images. One such popular recent approach is the U-Net architecture (Ronneberger, Fischer, and Brox, 2015), a fully convolutional network that has been used to achieve impressive results in the biomedical image domain. Unfortunately, existing segmentation methods require a significant amount of pixel-wise labeled training data, which can be difficult and slow to collect on novel domains. Given the importance of the segmentation problem in many domains, and due to the lack of supervised data for many problems, unsupervised segmentation appears to be the best solution.

One domain with many promising applications is food segmentation. For example, it can help estimate food size and hence, calories and analyze people's eating habits for healthcare. Although some work has been done related to food image recognition, there is not any relevant work which applies segmentation techniques using deep learning to this field.

The main objective of this project is to see if unsupervised segmentation using convolutional neural networks can be useful for semantic segmentation and more specifically, for food analysis, and compare it to other supervised and unsupervised segmentation algorithms.
In particular, we design our architecture inspired by the W-Net (Xia and Kulis, 2017), which ties two Fully Convolutional Network (FCN) architectures (each similar to the U-Net architecture) together into a single autoencoder. The first FCN encodes an input image, using fully convolutional layers, into a k-way soft segmentation. The second FCN reverses this process, going from the segmentation layer back to a reconstructed image. The idea given in (Xia and Kulis, 2017) is to jointly minimize both the reconstruction error of the autoencoder as well as a “soft” normalized cut loss function on the encoding layer. Instead of using the "soft" normalized cut loss function, we propose to use a faster alternative inspired on clustering, which tries to minimize the intra-class variance and to maximize the inter-class variance. In order to achieve better results, after computing the results for the W-net, we apply post-processing separating connected components in different classes. We compute the results for different variations of the W-net in order to obtain the best results and compare these results with other methods, such as the binary and the multi-class U-net (Ronneberger, Fischer, and Brox, 2015) or the supervised W-net. We tested our method on UNIMIB2015 Food Dataset, which contains 1027 tray images with multiple foods and 73 food categories.

The rest of this project is organized as follows. We first review related and relevant works in Chapter 2. Following this, in Chapter 3 we explain the methodology of our architecture. We also provide the definition of our *Centroid Loss* function, which we use analogously to the "soft" normalized cut loss in the classical W-Net. Chapter 4 presents the validation and results of the models and finally in Chapter 5 we discuss the conclusions drawn from this project.

The source code for all the work in this project can be found in
https://github.com/mbrufau7/tfm_food_segm.

Chapter 2

State of the art

In this Chapter we talk about the advances which have been made regarding segmentation, considering both supervised and unsupervised methods, and particularly, food segmentation.

2.1 Segmentation

During the long history of computer vision, one of the grand challenges has been studying and creating algorithms for segmentation, which is the ability to segment an unknown image into different parts and objects. There are many different approaches for this task. We show an overlook of the most important advances through the years.

2.1.1 First approaches, unsupervised segmentation

Traditional image segmentation algorithms are typically based on clustering, often with additional information from contours and edges. For example, in the simplest case, satellite image segmentation can often successfully be performed by clustering pixels based on wavelength, that is, one would create clusters based on similar pixels which are also located spatially nearby. The most important works can be seen in (D, R, and E, 2011), (M, V, and R, 2014) and (DE and PF, 2011).

There have been numerous enhancements and evolutions to the clustering approach. One of the most well-known and significant approaches is modeling using a Markov process (Geman, 1984). Another notable method is combining contour detection in a hierarchical approach (P et al., 2010). In SAR imagery, region growing with unsupervised learning is explored (P, AK, and DA, 2011). For good overviews of the older pre-deep learning approaches, we refer the reader to several surveys (DD and SG, 2013), (AA, SB, and N, 2011), (MW, 2014), (SR and E, 2012), (T et al., 2011), (R and M, 2011) which cover the works spanning color and edge image segmentation to medical image understanding. However, recent advances have made many of the older methods obsolete.

2.1.2 Recent advances, supervised segmentation

To be more precise (Mostajabi, Yadollahpour, and Shakhnarovich, 2014), (Farabet et al., 2012), (Dai, He, and Sun, 2014), (Hariharan et al., 2014b) and (Hariharan et al., 2014a) all make pixel-wise annotations for segmentation based on supervised classification using deep networks. Fully convolutional networks (FCNs) (Long, Shelhamer, and Darrell, 2014) have emerged as one of the most effective models for the semantic segmentation problem. In a FCN, fully connected layers of standard convolutional neural networks (CNNs) are transformed as convolution layers with kernels that cover the entire input region. By utilizing fully connected layers, the network can take an input of arbitrary size and produce a correspondingly-sized output map; for example, one can produce a pixel-wise prediction for images of arbitrary size. Recently a number of variants of FCN using segmentation have been proposed and studied, like (Noh, Hong, and Han, 2015), (Badrinarayanan, Kendall, and Cipolla, 2015), (Chaurasia and Culurciello, 2017), (Paszke et al., 2016), (Ronneberger, Fischer, and Brox, 2015), (Krähenbühl and Koltun, 2012) and (Zheng et al., 2015). In (Krähenbühl and Koltun, 2012), a conditional random field (CRF) is applied to the output map to fine-tune the segmentation.(Zheng et al., 2015) formulates a mean-field approximate inference for the CRF as a Recurrent Neural Network (CRFRNN), and then jointly optimize both the CRF energy as well as the supervised loss. (Guo et al., 2018) and (Thoma, 2016) provide comprehensive coverage of the top approaches and summarize the strengths, weaknesses and major challenges. Also, (Zhao et al., 2017), reviews different types of deep learning based fine-grained image classification approaches.

2.2 Food segmentation

Now we look more particularly at the works related to food segmentation. Food image recognition is one of the promising applications of visual object recognition, since it can help estimate food calories and analyze people's eating habits for healthcare. For images which contain two or more food items, segmentation of food is needed. Therefore, many unsupervised food segmentation papers have been published. (Matsuda, Hoashi, and Yanai, 2012) proposed to used multiple methods to detect food regions such as a deformable part model (DPM) (Felzenszwalb et al., 2009). (He et al., 2013) employed Local Variation (Felzenszwalb et al., 2011) to segment food regions for estimating total calories of foods in a given food photo. (Chen et al., 2015) follow a method based on a saliency-aware active contour model (ACM) (Shimoda and Yanai, 2015) propose a model which is first obtain region proposals by selective search, the estimate saliency maps and then apply GrabCut using the obtained saliency maps as seeds of GrabCut. In this project we obtain unsupervised food segmentation, in a unique approach, using a deep convolutional model, the W-net (Xia and Kulis, 2017). In addition, we compare this architecture with a supervised deep concolutional model, the U-net (Ronneberger, Fischer, and Brox, 2015), and with the supervised version of the W-net.

2.3 W-net

The model we are studying (Xia and Kulis, 2017) is a deep convolutional W-shaped architecture such that by taking an image as input it reconstructs the original input image and also predicts a segmentation map without any labeling information. This model is a modification of a supervised deep convolutional U-shaped architecture consisting of a contracting path to capture context and a symmetric expanding path that enables precise localization.

In this project, following the works of (Xia and Kulis, 2017), we design an encoder such that the input is mapped to a dense pixel-wise segmentation layer with the same spatial size rather than a low-dimensional space. The decoder then performs a reconstruction from the dense prediction layer. We compare the results with a supervised architecture such as U-net (Ronneberger, Fischer, and Brox, 2015).

Chapter 3

Methodology

In this chapter, we contain the main ideas of the networks we develop and test. First of all, we take a look at the works of (Ronneberger, Fischer, and Brox, 2015) for the U-net. Following this, we explore the main ideas of the W-net architecture presented in (Xia and Kulis, 2017). Finally, after looking into the main concepts for the "soft" normalized cut loss function proposed in the W-net paper, we propose a faster alternative based on clustering which we call *Centroid Loss* function.

3.1 U-net

The W-net ties two fully convolutional network architectures together into a single autoencoder. This two architectures are based on the U-net, which is a well known fully convolutional network (Ronneberger, Fischer, and Brox, 2015). It was presented in 2015 for Biomedical Image Segmentation on the ISBI challenge. The main idea of this network is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled output. A successive convolution layer can then learn to assemble a more precise output based on this information.

One important modification in this architecture is that the upsampling part has a large number of features channels, which allows the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a U-shaped architecture.

3.1.1 Architecture

The network architecture is illustrated in Figure 3.1. It consists of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. It repeatedly applies two 3x3 convolutions, each followed by a rectifier linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total, the network has 23 convolutional layers.

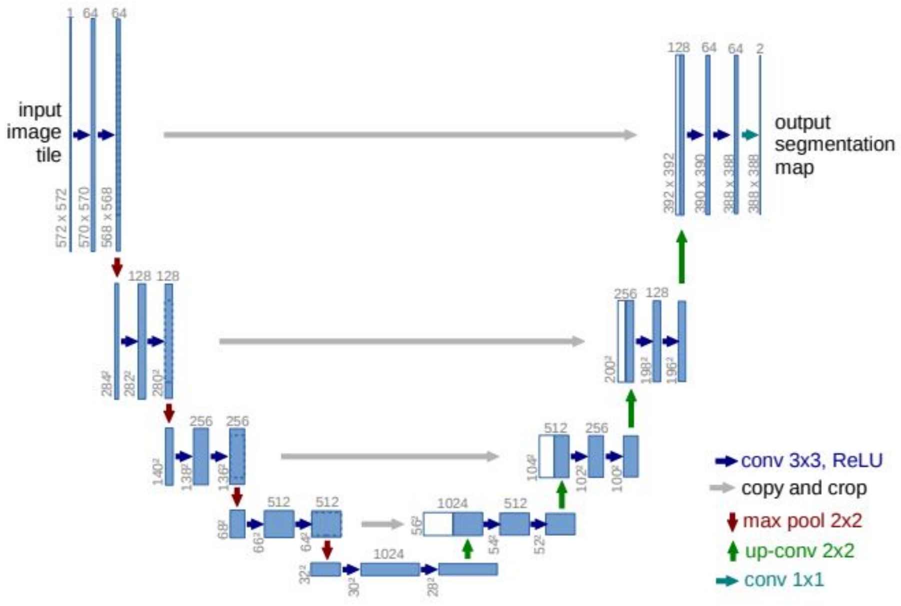


FIGURE 3.1: U-net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

3.1.2 U-net performance

In (Ronneberger, Fischer, and Brox, 2015), they apply the U-net to different segmentation tasks. The first task is the segmentation of neuronal structures in electron microscopic recordings. They used a set of 30 images for training, and achieved a warping error of 0.0003529 and a rand-error of 0.0382, which outperformed the other methods used for this problem by a large margin.
They also applied the U-net to a cell segmentation task in light microscopic images. The first data set "PhC-U373" contains 35 partially annotated training images. They achieved an IoU of 92%, which is significantly better that the second best algorithm with 83%. The second dataset "DIC-HeLa" to which they applied the U-net consists of 20 partially annotated training images. They obtained an average IoU of 77.5%, which again outperformed the second best algorithm with 46%.

3.2 W-net

In this project, our main goal is the unsupervised segmentation by deep learning. To this aim, we proceed with exploring in depth and attempting to improve the idea of the W-net (Xia and Kulis, 2017) as one of the most recent unsupervised segmentation techniques using convolutional neural networks. This idea was borrowed by the supervised segmentation methods of the paper (Ronneberger, Fischer, and Brox, 2015). To be more precise, the W-net achieves unsupervised segmentation by concatenating two fully convolutional networks (each similar to the U-net architecture) together into an autoencoder — one for encoding and one for decoding. The first fully convolutional networks encodes an input image, using fully convolutional layers, into a K-way soft segmentation. The second fully convolutional networks reverse this process, going from the segmentation layer back to a reconstructed image.

Both reconstruction error of the autoencoder as well as the error produced by the encoder are jointly minimized during training.

After obtaining an initial segmentation from the encoder, the authors performed two postprocessing steps, namely Conditional Random Fields (CRF) smoothing and hierarchical merging, in order to obtain the final result.

3.2.1 Architecture

The network architecture is illustrated in Figure 3.2. It is divided into an U_{Enc} (left side) and a corresponding U_{Dec} (right side). As we already mentioned, the authors modified and extended the typical U-shaped architecture of a U-Net network (Ronneberger, Fischer, and Brox, 2015). The symmetry observed between the encoder and the decoder creates a W-shaped architecture. This architecture reconstructs the original input images as well as predicts the segmentation maps without any labeling information. The W-Net architecture of the paper has 46 convolutional layers which are structured into 18 modules marked with the red rectangles. Each module consists of two 3 x 3 convolutional layers, each followed by a rectifier linear unit (ReLU) non-linearity and batch normalization. The first nine modules form the dense prediction base of the network and the second 9 correspond to the reconstruction decoder.

The U_{Enc} consists of a contracting path (the first half) to capture context and a corresponding expansive path (the second half) that enables precise localization, as in the original U-Net architecture. The contracting path starts with an initial module which performs convolution on input images. In Figure 3.2, the output sizes are reported for an example input image with resolution of 224 x 224. Modules are connected via 2 x 2 max-pooling layers, the feature channels are doubled at each downsampling step and halved at each upsampling step. In the expansive path, modules are connected via transposed 2D convolution layers. The final convolutional layer of the U_{Enc} is a 1 x 1 convolution followed by a softmax layer. The 1x1 convolution maps each 64-component feature vector to the desired number of classes K, and then the softmax layer rescales them so that the elements of the K-dimensional output lie in the range (0,1) and sum to 1. The architecture of the U_{Dec} is similar to the U_{Dec} except it reads the output of the U_{Enc} which has the size of 224 x 224 x K. The final convolutional layer of the U_{Dec} is a 1 x 1 convolution to map 64-component feature vector back to a reconstruction of original input.

One important modification in the W-net architecture compared to the U-net is that all of the modules use the depthwise separable convolution layers except modules 1, 9, 10, and 18. A depthwise separable convolution operation consists of a depthwise convolution and a pointwise convolution. The idea behind such an operation is to examine spatial correlations and cross-channel correlations independently—a depthwise convolution performs spatial convolutions independently over each channel and then a pointwise convolution projects the feature channels by the depthwise convolution onto a new channel space. As a consequence, the network gains performance more efficiently with the same number of parameters. In Figure 3.2, blue arrows represent convolution layers and red arrows indicate depth-wise separable convolutions.

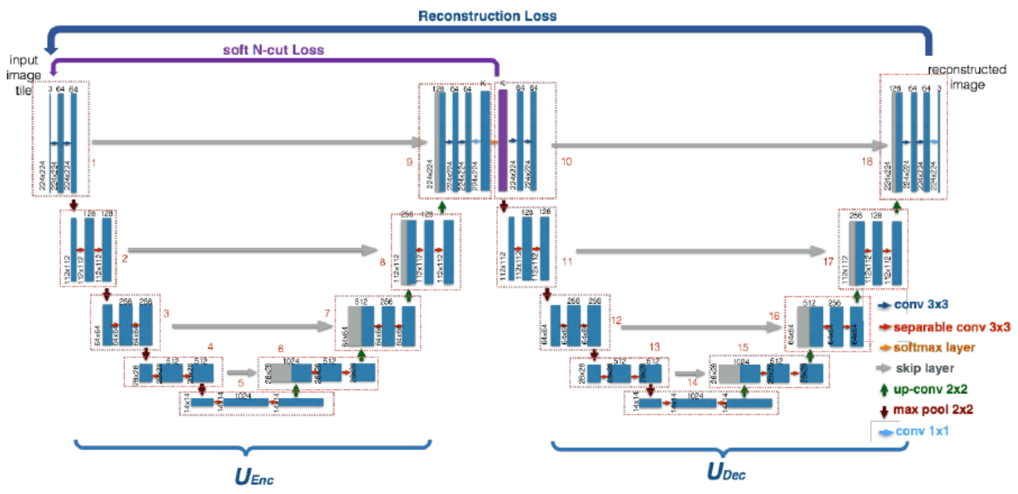


FIGURE 3.2: W-Net architecture: it consists of an U_{Enc} (left side) and a corresponding U_{Dec} (right side). It has 46 convolutional layers which are structured into 18 modules marked with the red rectangles. Each module consists of two 3x3 convolutional layers. The first nine modules form the dense prediction base of the network and the second 9 correspond to the reconstruction decoder.

3.2.2 W-net performance

In (Xia and Kulis, 2017), the W-net is tested in two Segmentation Datasets from Berkeley University, BSDS300 and BSDS500. The authors evaluated the performance on three different metrics: Variation of Information (VI), Probabilistic Rand Index (PRI), and Segmentation Covering (SC). For SC and PRI, higher scores are better; for VI, a lower score is better. In addition, they reported human performance on this data sets. The model outperforms a number of existing classical and recent unsupervised segmentation techniques, achieving performance near human level by some metrics.

3.3 Loss Functions of the Network

3.3.1 Soft Normalized Cut Loss

Introduction

We wish to segment the image into regions in a way that respects the structure of the image. For this we define a graph in which pixels are nodes, and weights $w(u,v)$ of the edges account for the similarity between the pixels they connect: it is a product of negative exponentials of both: the color distance

$$d_c^2(u,v) = ||RGB(u) - RGB(v)||^2$$

and the spatial distance

$$d_g(u,v)^2 = ||(i,j)_u - (i,j)_v||^2$$

between the pixel pairs (u, v):

$$w(u, v) = e^{-d_c^2/\sigma_c^2} e^{-d_g^2/\sigma_g^2}$$

scaled respectively by color and geometrical factors σ_c, σ_g

The idea behind the N-cut (not to be confused with Graph Min Cut) is to minimize, for each class, the ratio between:

- the similarities between pixels within the same class, and
- the similarities between pixels of the class and the other classes.

For this, let us define, for two disjoint sets of pixels A, B, $A \cap B = \emptyset$, the cut between them:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v),$$

that is, the sum of weights of connections between them. The associativity is the same concept, only that it applies for a subset A of V, $A \subset V$:

$$assoc(A, V) = \sum_{u \in A, v \in V} w(u, v),$$

that is, the connections from region A to the rest of pixels (including itself).
We first derive the loss function for the Binary N-Cut problem and later we will generalize it.

Binary N-cut problem

In (Shi and Malik, 2000), they realized that simply minimizing the cut between different regions produces very small regions.
For this reason and according to the notions we gave above, we define the normalized N-cut loss function of a partition of all the pixels in the image, V, into two subsets A, B as:

$$N_{cut}(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(B, A)}{assoc(B, V)},$$

which accounts for the separation between pairs of pixels of different classes, or disassociativity, normalized.
Instead, for measuring the normalized associativity, that is, the similarity of pixels between the same class, the authors define:

$$N_{assoc}(A, B) = \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)}$$

Lemma 3.3.1. *For the binary problem, we have the following equivalence:*

$$N_{cut}(A, B) = 2 - N_{assoc}(A, B)$$

Proof. First note that since $B = V - A$, the following is valid:

$$cut(A, B) = assoc(A, V) - assoc(A, A)$$

Applying this to the N_{cut} definition, we get:

$$\begin{aligned}
N_{cut}(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)} \\
&= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\
&= 2 - \frac{assoc(A,A)}{assoc(A,V)} - \frac{assoc(B,B)}{assoc(B,V)} \\
&= 2 - N_{assoc}(A,B).
\end{aligned}$$

□

N-cut loss for the non binary problem

For a segmentation of an image into $K > 2$ classes, we can generalize the above observation and define the Normalized N-cut Loss:

$$N_{cut}(A_1,\dots,A_K) = \sum_{k=1}^{K} \frac{cut(A_k, V - A_k)}{assoc(A_k,V)} = K - \sum_{k=1}^{K} \frac{assoc(A_k,A_k)}{assoc(A_k,V)}.$$

However, this definition only applies for a "hard" segmentation, that is, the assignment of a class to each pixel. This does not yield a differentiable function, which is what we need for our optimization problem.

Soft N-cut loss

In (Xia and Kulis, 2017), the authors devise a solution: assign a membership vector to each pixel with as many components as classes we have, indicating the probabilities of belonging to each class.
By replacing the hard constraints of belonging to a class inside the sums across the pixels by a probability weighting each term, they obtain the formula for the Soft N-cut Loss:

$$\mathcal{L}(A_1,\dots,A_K) = K - \sum_{k=1}^{K} \frac{\sum_{u,v} p(u \in A_k)p(v \in A_k)w(u,v)}{\sum_{u,v} p(u \in A_k)w(u,v)},$$

where in the denominator we directly omit the term $p(v \in V) = 1$.
The image segmentation should now look like a 3D box with depth K, and at each depth level k we should find the image of the probabilities of each pixel belonging to class k, taking continuous values between 0 and 1.

Technical Optimization for the Soft N-cut loss

The above formula would involve a lot of unnecessary computational cost as the number of pairs of pixels grows as $(m \times n)^2$ with the width m and height n of the picture.

1. Neighbours only:

   In the W-net, only connections between neighbour pixels are considered to have non-zero similarity (non infinite distance). It is achieved by thresholding the similarity between pixels to radius $r = 5$ with a factor:

$$w_r(u,v) = e^{-d_c^2/\sigma_c^2} e^{-d_g^2/\sigma_g^2} \cdot \begin{cases} 1, & \text{if } d_g(u,v) \leq r \\ 0, & \text{otherwise} \end{cases}$$

2. Vectorizing:

   For speeding up calculations and also compactness of the formula, we observed that the former can be expressed as matrix and vector products.

   - Let $\mathbf{p_k} = (p(u_1 \in A_k), \ldots, p(u_N \in A_k))$ be the vector of probabilities of each pixel $1, \ldots, N = n \cdot m$ belonging to class k
   - Let $(W)_{u,v} = w(u,v)$ be the matrix of weights
   - Let $\mathbf{1}$ be a vector of ones with N dimension

   then the loss function can be simply thought as

$$\mathcal{L}(A_1, \ldots, A_n) = K - \sum_{k=1}^{K} \frac{\mathbf{p_k}^T W \mathbf{p_k}}{\mathbf{p_k}^T W \mathbf{1}}$$

   This observation reveals W as the metrics matrix of the space of segmentation. By restricting only to neighbouring pixels, we obtain a somewhat banded-diagonal sparse W matrix (not exactly tri-diagonal because neighbouring pixels in the original image are not always neighbors after reshaping it into a vector).

3. Storing the weights:

   Another step to reduce the computational effort of such a loss function in our implementation of *Keras* using *TensorFlow* as backend was to store the weights of the similarity matrix, which only depends on the original image and not on the current state of the segmentation being optimized inside the training process.

   Since the current interface of *Keras* allows for custom defined loss functions for the Neural Network as long as they take two arguments, ground truth and prediction and return a loss value, we needed to pack the values of the weights into a tensor that matched in shape the dimension of images. For that, we took the approach of attaching into the images, on top of the 3 layers corresponding to R,G and B values, one layer for each pixel in the connectivity pattern.

   For example, we store the similarity between every pixel of the image and the pixel to its right in the first added layer, and so on for all the neighbours below radius 5. This way we do not waste any extra space, and at loss evaluation time, we unwrap it to get the sparse W matrix.

3.3.2 Fast alternatives inspired on clustering

Centroid loss

Despite the efforts put on optimizing the *Soft N-cut Loss* implementation, it takes a lot of time for *TensorFlow* to compile, and only finishes for very small images (32×32). For this reason, we propose a loss function that involves less pixel-to-pixel comparisons, but preserving the common idea of minimizing the **intra-class variance** and maximizing the **inter-class variance** (usually minimizing the ratio of the former and the latter). In our case, where we only have the notion of similarity, we propose to

maximize intra-class similarity and to minimize inter-class similarity.
We can rethink the segmentation problem as a classification problem on a pixel level. We want to assign to each pixel:

- a class label for the "hard" segmentation problem $y(x) \in \{1, \dots, K\}$
- or at least a membership vector along all classes for the "soft" segmentation problem $y(x) = (y_1, \dots, y_K) = (p(x \in A_1), \dots, p(x \in A_K))$,

based on the 5 available features:

- RGB color, and
- (i, j) pixel coordinates.

So pixels are points on a $5-$ dimensional feature space to which we can apply some classical clustering approach to obtain a differentiable measure of the goodness of a classification into K classes.
We arrange data in the typical form of $(X|y)$ one row per sample and one column per feature or label:

$$(R(x), G(x), B(x), I(x), J(x) | p_1(x), \dots, p_k(x)).$$

There are two very basic descriptors within a classification process. For each class k, we can compute very fast:

- The centroid of class k, i.e. the mean of points classified as k

  $$\boldsymbol{\mu}_k(X, \mathbf{y}) = \boldsymbol{\mu}(A_k) = \sum_{\mathbf{x} \in A_k} \frac{\mathbf{x}}{n_k}$$

  for $n_k = \#A_k \neq 0$ the cardinal of $A_k = \{\mathbf{x} | y(\mathbf{x}) = k\}$ in case it is non-zero.

  For the case of a soft segmentation

  $$\boldsymbol{\mu}_k(X, y) = \frac{\sum_{\mathbf{x}} p(\mathbf{x} \in A_k)\mathbf{x}}{\sum_{\mathbf{x}} p(\mathbf{x} \in A_k)} = \frac{\mathbf{p_k}^T X}{\mathbf{p_k}^T \mathbf{1}}.$$

  Again, each class needs to have a non-zero accumulated probability along pixels in order for the centroid to be defined.

  Note that according to the strict definition of centroid of a set of points in a non-necessarily Euclidean space, that is, the point that minimizes the sum of distances to each point of the set, the mean does not necessarily coincide with it. But since in our space, the distance is only affected by a scaling of the color component by σ_c and the geometrical component by σ_g, we can make it Euclidean after rescaling the coordinates. If we apply the "neighbours only" thresholding to the distance, this is not true for non-neighbour pixels.

  With this already, we can get a loss function from the sum of similarities between centroids, but we can go further.

- The variance of class k, i.e. the mean of the squared distance of points classified as k to the class centroid:

$$\sigma_k^2(X,y) = \sigma^2(A_k) = \sum_{\mathbf{x}\in A_k} \frac{(\mathbf{x}-\boldsymbol{\mu}_k)^2}{n_k}.$$

Again, for the soft segmentation, we replace the belonging condition by weights:

$$\sigma_k^2(X,y) = \frac{\sum\limits_{\mathbf{x}} p(\mathbf{x}\in A_k)(\mathbf{x}-\boldsymbol{\mu}_\mathbf{k})^2}{\sum\limits_{\mathbf{x}} p(\mathbf{x}\in A_k)} = \frac{\mathbf{p_k}^T(X-\boldsymbol{\mu}_k)^2}{\mathbf{p_k}^T\mathbf{1}},$$

where the ^{2} operator denotes self element-wise product.

Note that $\sigma_k^2(X,y)$ is a vector of the variance of each feature across samples, weighted by the class probability. To get a scalar value, we can take the sum, weighted by the corresponding color and position weights σ_c, σ_g.

From the centroids and the variances, we can build an equivalent definition of the *N-cut Loss* or its soft version, but with much less pixel comparisons.

- Replace the sum of similarities between pixels of the same class:

  $$\sum_{A_k} e^{-d(u,v)^2}$$

  by the negative exponential of the variance scalar value

  $$\sigma_k^2 = RGB(\sigma_k^2)/\sigma_c^2 + IJ(\sigma_k^2)/\sigma_g^2$$

  , that is, by

  $$e^{-\sigma_k^2} = w(\boldsymbol{\sigma_k}, \mathbf{0}).$$

  The order of summing and exponentiating affects the result, but the scale should be the same since the variance is the mean of the squared distance to the centroid, and the concept being penalized is the same.

  We have seen that the negative exponential of the magnitude of the variance can be rewritten as the pixel similarity between the standard deviation vector and the zero vector for simplicity.

- Replace the sum of similarities between pixels of two different classes, k_1, k_2 by the negative exponential of the squared distance between its centroids, that is, just the pixel similarity between the centroids:

  $$e^{-d(\boldsymbol{\mu}_{k_1},\boldsymbol{\mu}_{k_2})^2} = w(\boldsymbol{\mu}_{k_1}, \boldsymbol{\mu}_{k_2})$$

  That is, instead of taking the mean, or sum, of the negative exponential of the distances, we take the negative exponential of the distance of the class means, so we only compare two points.

The formula for the loss function following this approach is:

$$\mathcal{L}_{centroids}(A_1,\ldots,A_K) = K - \sum_{k=1}^{K} \frac{w(\boldsymbol{\sigma_k}, \mathbf{0})}{\sum\limits_{i,j} w(\boldsymbol{\mu}_{k_1}, \boldsymbol{\mu}_{k_2})}.$$

We can think of this approach as building the similarity matrix only on the centroid points, and replacing the diagonals (self similarity) with the (negative exponential of the) variances.
However, in the *Soft N-cut Formula*, we can see that similarities between pixels of the same class also appear in the denominator, so we finally add the term accounting for intra-class similarities from the numerator into the denominator:

$$\mathcal{L}_{centroids}(A_1, \ldots, A_K) = K - \sum_{k=1}^{K} \frac{w(\sigma_k, \mathbf{0})}{w(\sigma_k, \mathbf{0}) + \sum_{i,j} w(\boldsymbol{\mu}_{k_1}, \boldsymbol{\mu}_{k_2})}.$$

This also makes the fractions bounded in $[0, 1]$ so the loss function is positive.

Regularizing the segmentation by Penalizing bad shapes

Although the algorithm is unsupervised, we can give it some bias towards some good shapes for the regions it converges to. This fast alternative produces clouds of points around the class centroids, but sometimes it is not possible to get a good split and smooth boundary in the position coordinates given the high variation of colour at pixel scale from shapes, illumination, etc.
For this reason, we will add a term to the loss function that accounts for the **perimeter** of the shapes in (I, J) of the regions that correspond to a segmentation.
Let P be an image of the same width and height as the original image and the probabilities of belonging to each class as channels, or layers.
We take the convolution of the layer $P[:, :, k]$ by a gradient filter like:

$$C = \begin{pmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{pmatrix}.$$

The result is an image with high values where the regions defined by that class have edges in either direction. We take the sum of all values of the layer convolution, normalizing by the sum of values $P^k = P[:, :, k]$ so that layers with small regions contribute the same. We finally sum up across layers. So in brief, the perimeter loss is

$$\sum_{k=1}^{K} \frac{\sum_{i,j} (P^k \circledast C)_{ij}}{\sum_{i,j} P^k_{ij}}.$$

3.3.3 Reconstruction Loss

As in any classical encoder-decoder architecture, we also train the W-Net to minimize the reconstruction loss to enforce the encoded representation to contains as much information of the original inputs as possible. By minimizing the reconstruction loss, we can make the segmentation prediction align better with the input images. To achieve this, we tested two different loss functions:

- Mean Squared Error,
- Binary Cross-entropy.

From this two losses Binary Cross-entropy performs better for our models.

3.4 Post-processing and Coloring

To better visualize a segmentation, especially in the unsupervised scenario, we assign to each class the mean color of the pixels belonging to it. We will refer to this as the *hard mean color*. We will also plot separately for each class, the pixels that get assigned the class label, in binary black and white format, and we refer to them as *winners* of the class.
For the case of a soft, or k-way segmentation, the mean can be weighted by the class probability of the pixels, to which we will refer as the *soft mean color*. We will also plot separately for each class the probabilities of membership to the class as grayscale images.
The *soft mean color* visualization is more faithful to the *centroid loss* neural network, since it reflects the true color of the centroids being optimized. However, it has the drawback that probabilities of a pixel to be in a class that are non-zero but are not the maximum for the pixel, will make the mean color "dirty" for very wide classes that mix heterogeneous pixels. Also, the reconstruction loss gets information from these non-zero, but non-maximum probabilities to reconstruct the image, so these will not fully disappear in the W-net. Hence, we observe somewhat indistinguishable colors for adjacent classes.
If instead we convert the soft segmentation into a hard segmentation by taking the maximum to be 1 and zero-ing out the rest of the classes of a pixel, we can get a *hard mean coloring*, with much more vivid colors.
Finally, in our context of food, we do not wish classes to have more than one connected component, something that we observe frequently, and is against the assumption that the menus do not contain repeated dishes.
For this, we apply a very simple step of **post-processing** that consists on splitting each class obtained from a hard segmentation into its connected components, always discarding the background as a class. From this extended segmentation, we will again compute the *hard mean color* of each new class. We expect this to exhibit much more clearly defined colors.
Pay attention though on the initial state of training, where this visualization can look misleadingly good if the initial classes have hundreds of connected components. In this case, the resulting new regions will be very small, each one with a very similar color to the original image.

Chapter 4

Validation and Results

4.1 Datasets

For testing the models implemented in our project, we use the UNIMIB2016 Food Database (Ciocca, Napoletano, and Schettini, 2017). This database was originally designed for automatic dietary monitoring of canteen customers based on robust computer vision techniques. It consists of 1027 tray images with multiple foods and containing 73 food categories. You can see an example of an image from this dataset in Figure 4.1 and look at all food classes in Figure 4.2. The dataset contains segmentation masks for each image and it provides a split to use for train and testing with 650 images for training and 360 images for testing.

The different classes appear in different frequencies in the train and test set. For example, the most frequent class in the train set is "pain". It appears with a frequency of 40.77% in the train set and 43.06% on the test set.
On another hand, the least frequent classes in the training set are "pasta e ceci" and "torta crema". "Pasta e ceci" appears with a frequency of 0.46% in the train set and 0.56% on the test set, and "torta crema" appears with a frequency of 0.31% in the train set and 0.83% in the test set. As we can see by this statistics, the different classes are very unbalanced in the UNIMIB dataset.

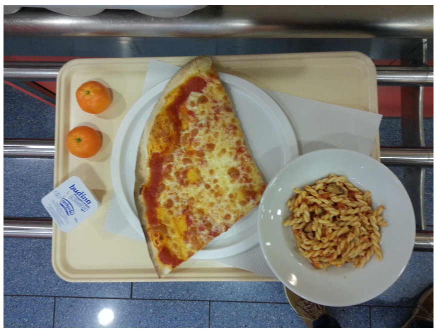


FIGURE 4.1: Example of an image from the UNIMIB2016 Dataset

4.1.1 Preprocessing

The ground truth for this dataset is given in a Matlab map structure, with the following format. Each entry in the map corresponds to the annotation of an image and each entry contains cell tuples as annotated food. A tuple is composed of 8 cells with the annotated:

- Item category (food for all tuples)

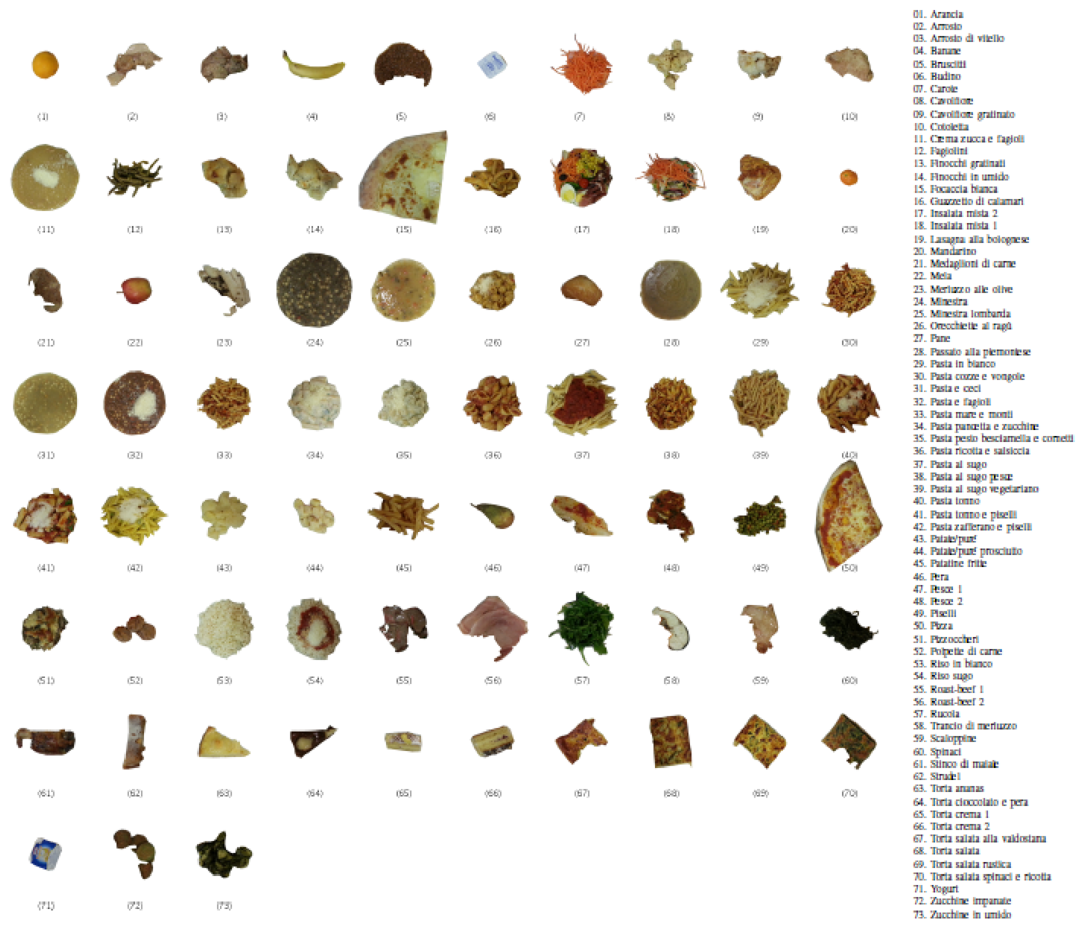


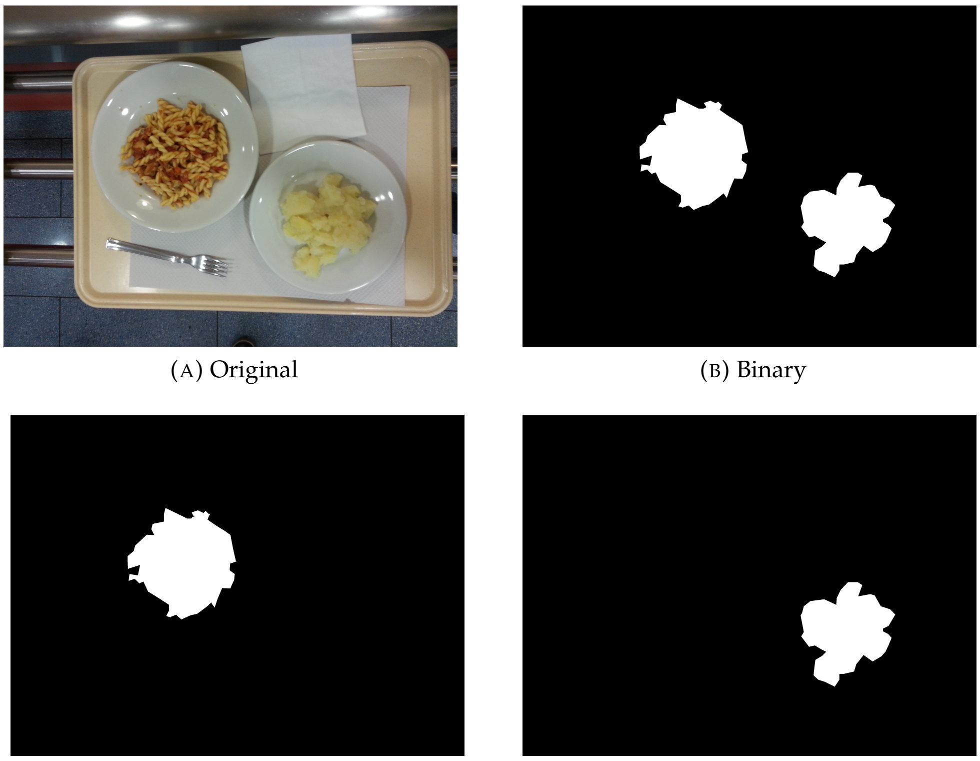
FIGURE 4.2: Food classes from the UNIMIB2016 Dataset

- Item class (e.g. pasta, patate, ...)
- Item name
- Boundary type (polygonal for all tuples)
- Item's boundary points $[x_1, y_1, x_2, y_2, ..., x_n, y_n]$
- Item's bounding box $[x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4]$.

In order to be able to use the ground truth in our project, we transform this map structure into matrices with the corresponding segmentation. As we want to make different tests with the ground truth, we create two different types of ground truth. We create binary ground truth with only two classes corresponding to the pixel belonging to food or to background. We also generated matrices for every class in order to be able to train the model with the different 73 classes. We stored this matrices as .png images. You can see an example for the ground truth in Figure 4.3

4.1.2 Data augmentation

To improve the results for the segmentation architecture, we perform some data augmentation to have more images for training.


(A) Original (B) Binary

(C) Pasta mare e monti (D) Patate pure

FIGURE 4.3: Ground truth for the UNIMIB2016 Dataset

The first form of data augmentation we use, consists of generating horizontal reflections. The second form of data augmentation consists of altering the intensities of the RGB channels performing PCA on the set of RGB pixel values throughout the training set. To each training image, we add multiples of the found principal components, with magnitudes proportional to the corresponding eigenvalues times a random variable drawn from a Gaussian with mean zero and standard deviation 0.1. Therefore, to each RGB image pixel $I_{xy} = [I^R_{xy}, I^G_{xy}, I^B_{xy}]$, we add the following quantity:

$$[\vec{p}_1, \vec{p}_2, \vec{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T,$$

where $\vec{p}_i$ and λ_i are the i-th eigenvector and eigenvalue of the 3×3 covariance matrix of RGB pixel values, respectively, and α_i is the aforementioned random variable. This form of data augmentation captures an important property of natural images, that object identification is invariant to changes in the intensity and color of the illumination. We can see an example of the PCA data augmentation for our dataset in Figure 4.4

4.2 Implementation details

We run all the experiments using Google Colaboratory. This tool gives up to 12 hours of uninterrupted execution with available Tesla K80 GPU and 12GB of RAM. As some of our architectures needed great computational power in order to run, this tool provides a comfortable environment for developing. We developed our models using Keras with Tensorflow as backend.

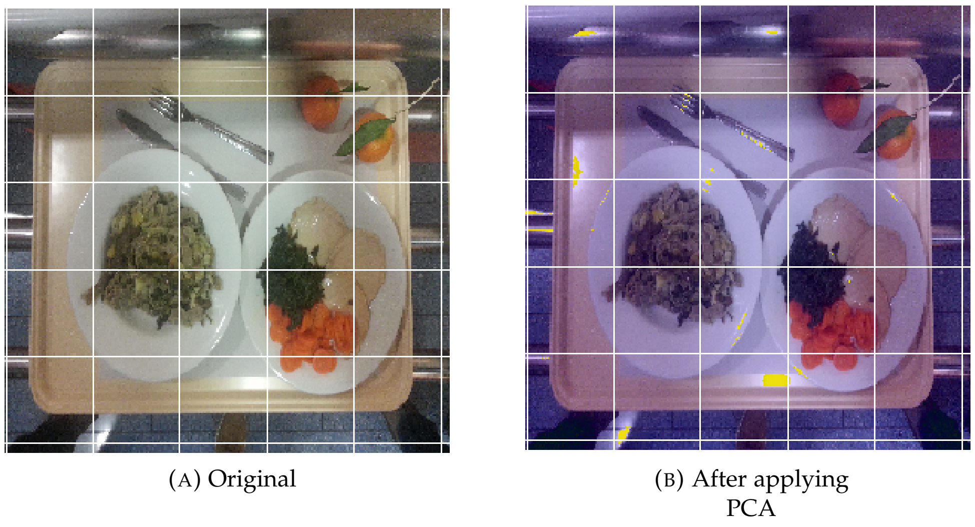

(A) Original (B) After applying PCA

FIGURE 4.4: Example of UNIMIB2016 image after applying PCA

4.3 Metrics

In order to evaluate the different models, we made two types of evaluation. First, we compute the mean Intersection over Union (IoU) for the test set and then we make a visual evaluation of the results.

4.3.1 Intersection over union

The intersection over union of a pair of regions measures the ratio between intersecting area and union area. It is useful for assessing how similar are the region of pixels that are known to correspond to a class and the region of pixels that are predicted to be in that class.
We compute this metric in two different ways for the supervised and unsupervised architectures. On the one hand, for the supervised setting, for each image, we just computed the mean across different classes of IoU between ground truth regions and predicted regions. Then, we computed the mean of these values across the whole test split.
On the other hand, for the unsupervised architecture, since the predicted class regions may not appear in the same order as the ground truth class regions, for each image, we computed the IoU between each class region from the ground truth and each class region given by the segmentation. We kept the best result for each class appearing in the ground truth and computed the mean between these values. Finally, we computed the mean for all test images as in the supervised setting.

4.3.2 Visual evaluation

We want to have a perception of the goodness of the segmentation, so we computed different visualizations. First of all, for the unsupervised setting, we make a plot with different features. We show the original image, the segmentation obtained painted with the mean color for each class, the segmentation obtained painted with the mean color after post-processing and the reconstruction image. We also show the probabilities for each class in the form of a grey scale image and the hard segmentation for each class with a black and white image, where the black pixels are the ones inside the class. An example of this visualization is available in Figure 4.5. Then, for both supervised and unsupervised results, we computed the contours of each class and compared them with the contours of the ground truth. An example of this visualization is available in Figure 4.6.

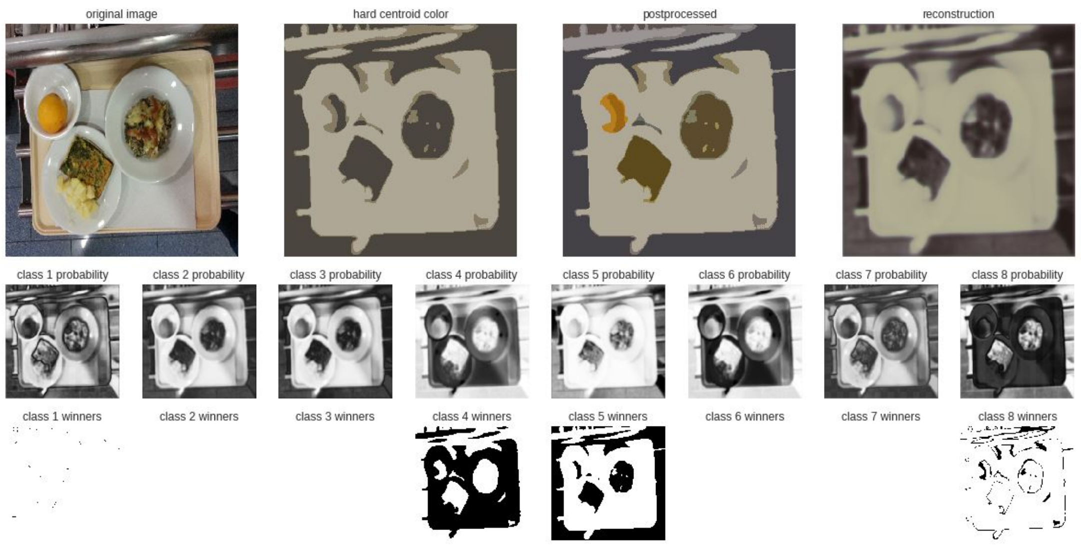


FIGURE 4.5: Example of unsupervised visualization

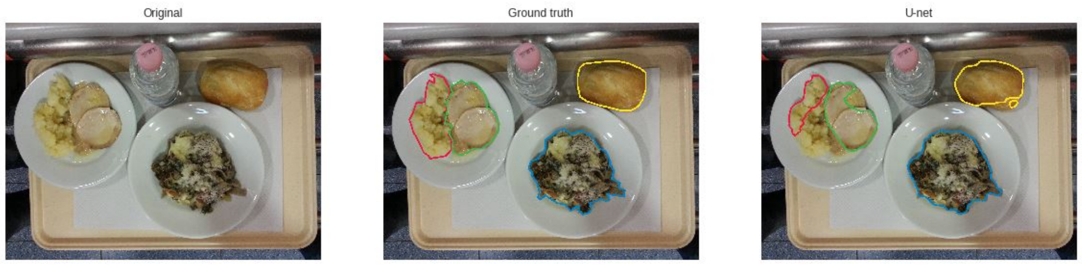


FIGURE 4.6: Example of contour visualization

4.4 U-net architecture

In order to make some comparison between supervised and unsupervised segmentation, we explore an architecture for the U-net. We take an already developed architecture from the DATA-SCIENCE-BOWL-2018 competition on kaggle (`https://github.com/kamalkraj/DATA-SCIENCE-BOWL-2018`) and we adapt it for our data.

The network has 9 modules and starts with 16 feature channels in the first convolution. We used 0.1 dropout in the first and the last 2 modules and 0.2 dropout on the rest of the modules. Also, this model had *Adam* as an optimizer and *He normal* as a kernel initialization. The network has 1.9 million parameters.
We test the network in two ways. The first one is using binary ground truth, i.e., we use only two classes for the pixels belonging to food or background. On the other hand, we test the network with the 73 food classes.

4.4.1 Binary U-net

We test different optimizers for this architecture and find that the one that gives the best results is the *Adamax* optimizer (see Table 4.1 for all the results).

Optimizer	IoU
`Adam`	0.85
`SGD`	0.62
`Adagrad`	0.14
`Adadelta`	0.79
`Adamax`	**0.87**
`Nadam`	0.14

TABLE 4.1: Results for the U-net with different optimizers

We compute the contour plots for the results coming from *Adam* in Figure 4.7. As we can see, the results are quite good, although the segmentation misses to give a good result for the "yogurt" class. The other classes are well segmented, and we can see it fails on recognizing pointy shapes, as ecidenced with the class "carotte" in the first image.

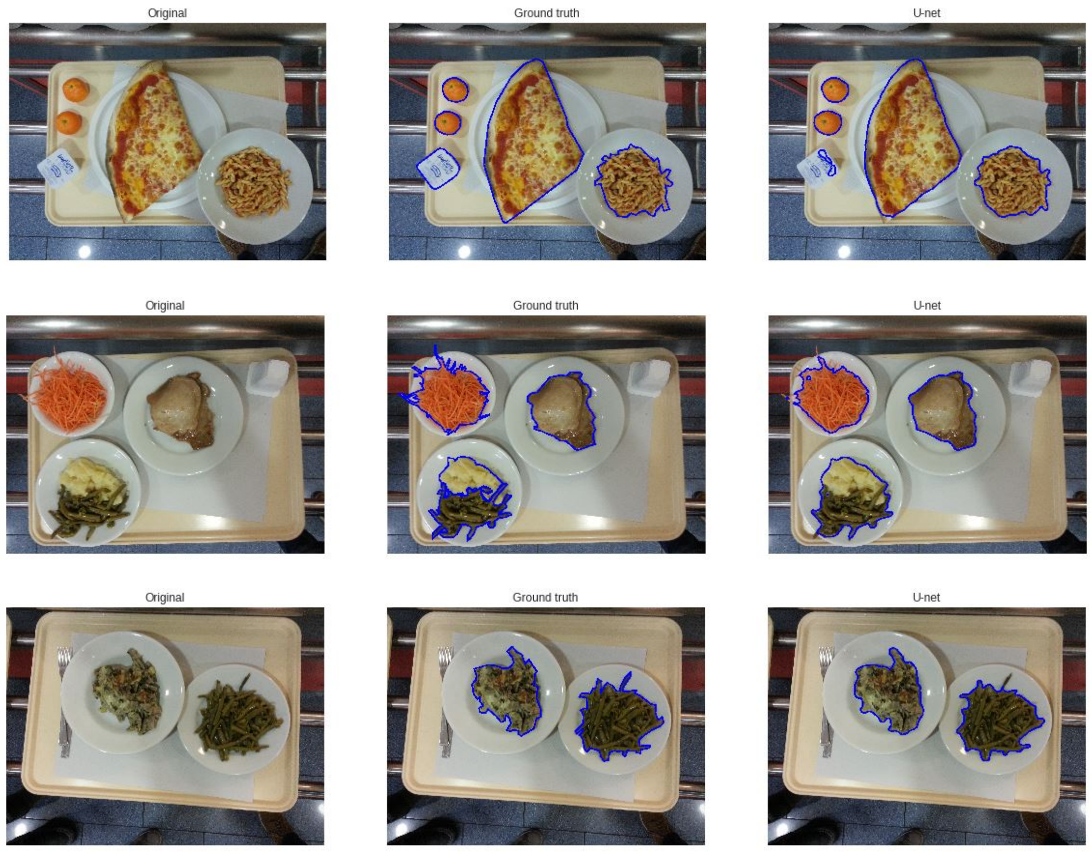


FIGURE 4.7: Results for the U-net with binary segmentation

4.4.2 Multiclass U-net

We test this architecture with the *Adamax* optimizer (which gives the best results for the binary problem). The result of the IoU for this architecture with *Adamax* is **0.65**. We see that these are much worse results than the binary problem, although it is expected, as here we have 73 classes. The visualization for these results is available in Figure 4.8. As you can see, in this case the network misses some classes like the "yogurt" and the "cotoletta" class.

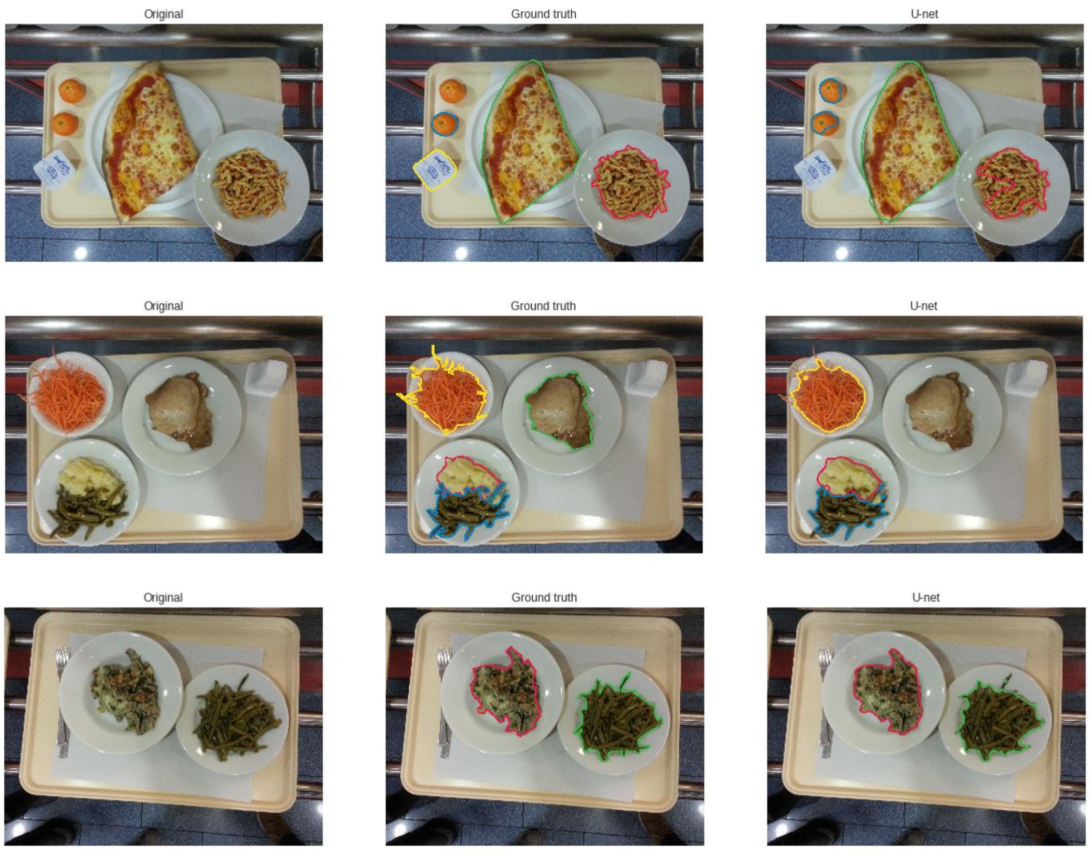


FIGURE 4.8: Results for the U-net with multiclass segmentation

4.5 Classical N-cut

For the Classical N-cut, we use the *scikit-learn* segmentation module to build the similarity graph, see (Shi and Malik, 2000), not from pixels directly, but from clustered regions of similar position and color.
The similarity matrix is only built on a few hundred nodes after having merged pixels into regions.
The results in 4.9 are pretty good, except that perhaps they do not merge sub-regions of the same food class due to color variations.
The evaluation of the **N-cut loss** on the **N-cut segmentation** (around 0.005) is much smaller than evaluating the**N-cut loss** on the **ground truth** (around 0.3), since the N-cut segmentation minimizes the N-cut loss.
This tells us that the N-cut loss does not exactly correspond to the semantic loss.

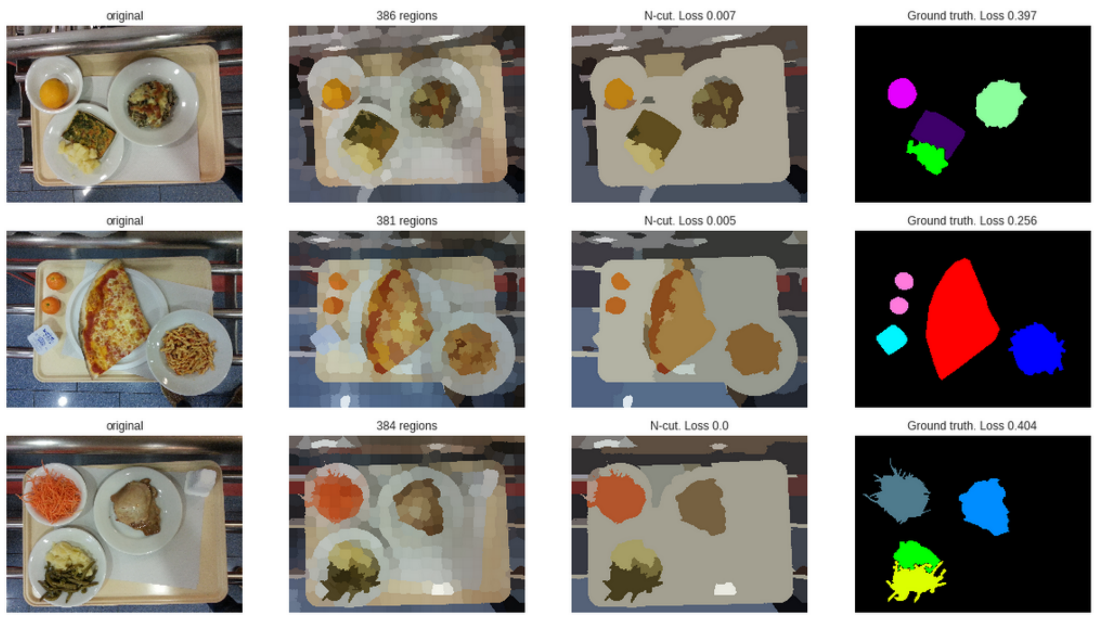


FIGURE 4.9: Results for the Classical N-cut

4.6 Embedding architecture

Before plugging the loss functions, that we have iteratively improved, directly on the W-net, we follow the following approach.
We take only one image, and a very simple neural network consisting of an Embedding layer from Keras.
An embedding layer takes a list of integer id's and maps a vector to each. The components of the vectors are the parameters of the layer.
So we will pass it as input a vector of id's $(0, ..., n_{rows} \cdot n_{cols} - 1)$ representing all the pixels, and the network will output (predict) a vector of membership probabilities for each one. Since there is no ground truth, we will pass the original image to the loss function so that it can compute a value from both original image and predicted probabilities.
The loss function (unsupervised) will propagate directly from the original image into the probabilities of the k-way segmentation, so this approach will get rid of the effect of the convolutional layers having to transmit information between the original image and the segmentation.
In the results, see Figure 4.10, where we use the *Centroid Loss*, we don't observe as much the effect that we see in the W-net that some classes tend to disappear very often, but rather the opposite, small shapes tend to stay too much along training and *oversegment* the image. Looking at the mean colors gives **an intuition about the effect of the Centroid Loss function isolated from the convolutional layers and the reconstruction loss**.
The IoU of the images we choose for evaluating is 0.4 for the pizza image, 0.45 for the carrots image (shown here) and 0.66 for the beans image, so in average it is 0.5 IoU.

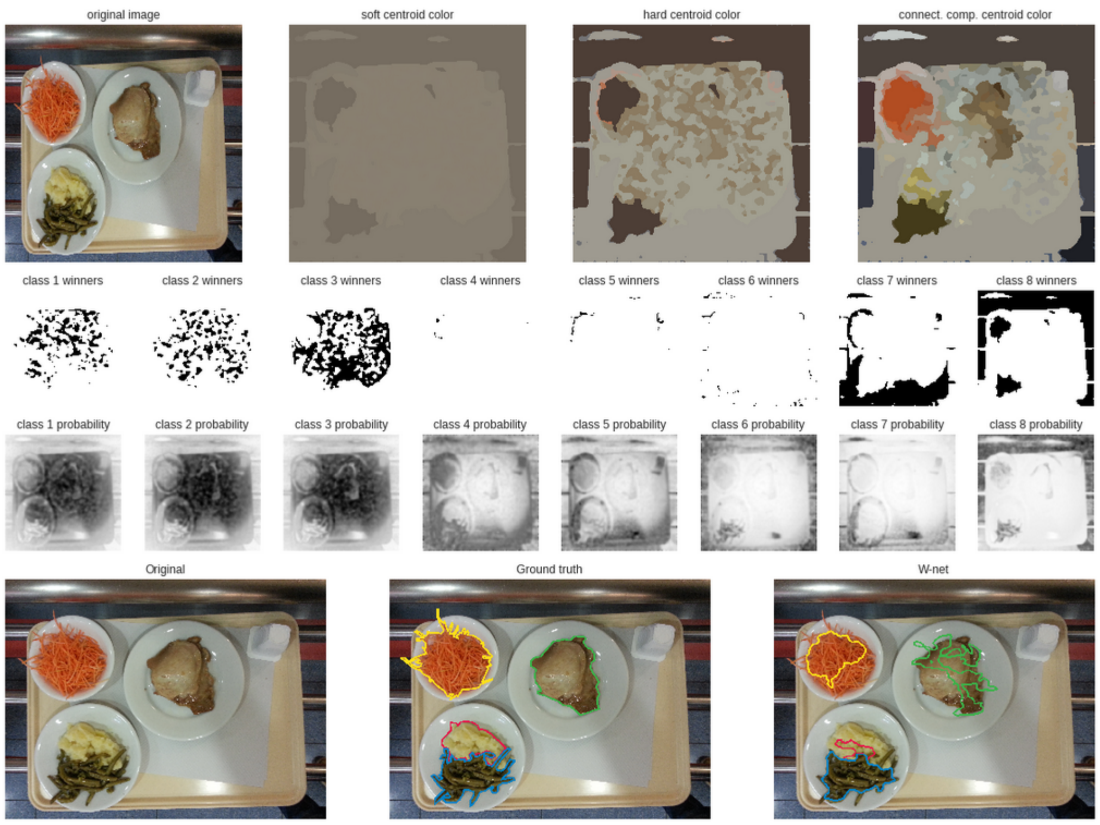


FIGURE 4.10: Results for the Embedding; 0.45 IoU

4.7 W-net architecture

In order to achieve the best segmentation possible, we create and compare four different architectures by which we get their names depending on the number of modules and the number of feature channels the image maps in the first convolution:

- W-net 18-64,
- W-net 18-32,
- W-net 14-64,
- W-net 10-64.

The three last architectures are modifications of W-net 18-64 which is the model used in (Xia and Kulis, 2017). The modifications made are concerning the number of modules and the number of feature channels at each convolution.
First, we use Adam as the optimizer for these architectures and He normal as a kernel initializer which were both used for the U-net implementation. Once we have obtained which of these to get the best results, we test the best network with different optimizers and initializers in order to confirm the best results.

4.7.1 W-net 18-64

This architecture was already analyzed in depth in Section 3.2.1. This model has 12,225,542 parameters and the dropout used for training is 0.65.

The mean IoU for this architecture is **0.29**. If we look at the results for the visualization 4.11, we can see that the segmentation is poor and this was one of the main reasons why we tried different architectures. Looking at both the results after postprocessing and the contour, we can see that all three food classes are not well segmented. Even the detection of the class "carote" which can be considered the easiest food of this image, since it has a standard colour, was segmented very inaccurately.

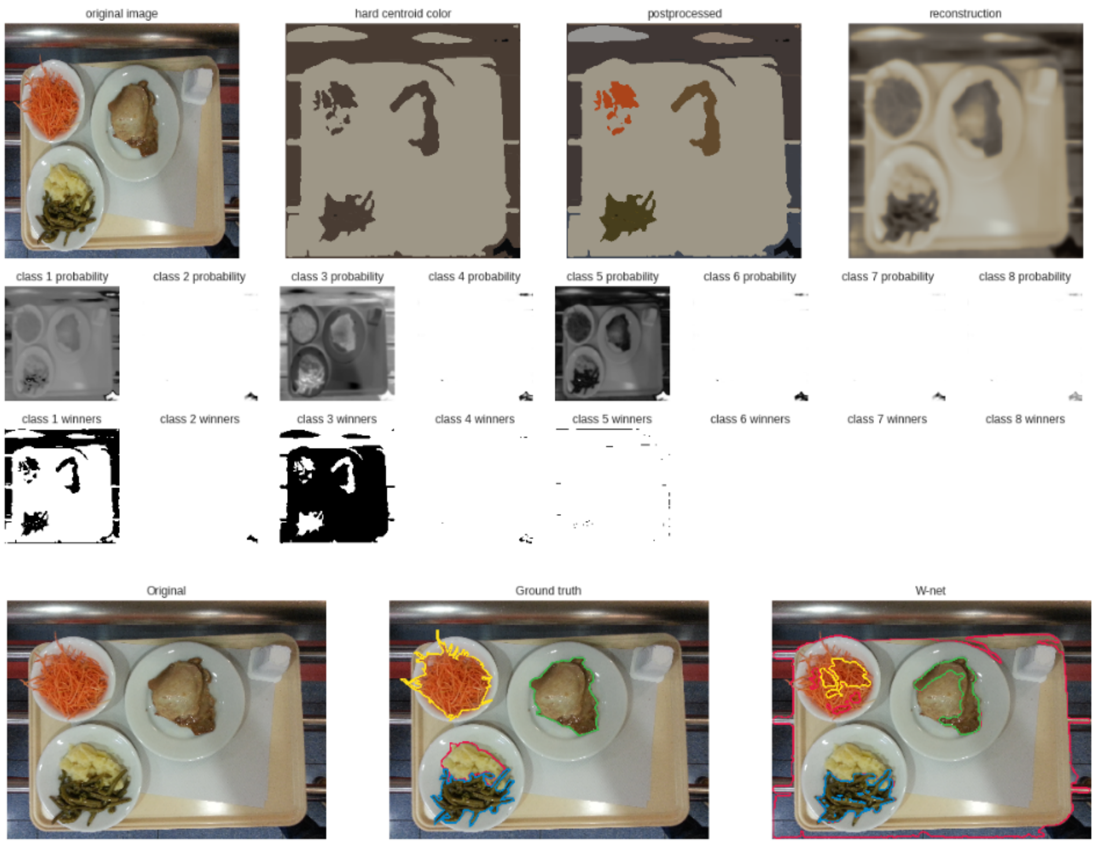


FIGURE 4.11: Results for the W-net 18-64

4.7.2 W-net 18-32

The first modification we make of the W-net 18-64, is decreasing the number of feature maps for each convolution. By decreasing the number of feature maps at each convolution by half, we obtain 3,085,830 parameters. In addition, the dropout used for this architecture is decreased from 0.65 to 0.2.

The mean IoU for this architecture is **0.52**. Looking at the results obtained with visualization, see Figure 4.12, we can see that we obtain 4 different hard segmentation classes, but that the food is mainly in the first class. After postprocessing, we obtain a quite good segmentation for all the food, except for the "patate/puré" class, which has gotten mixed with the background, and as we can see in the contour plot, this class does not have a good segmentation.
This model obtains the best results when it comes to the unsupervised W-net with Centroid loss as the loss computed from the encoder and binary cross-entropy used as reconstruction loss.

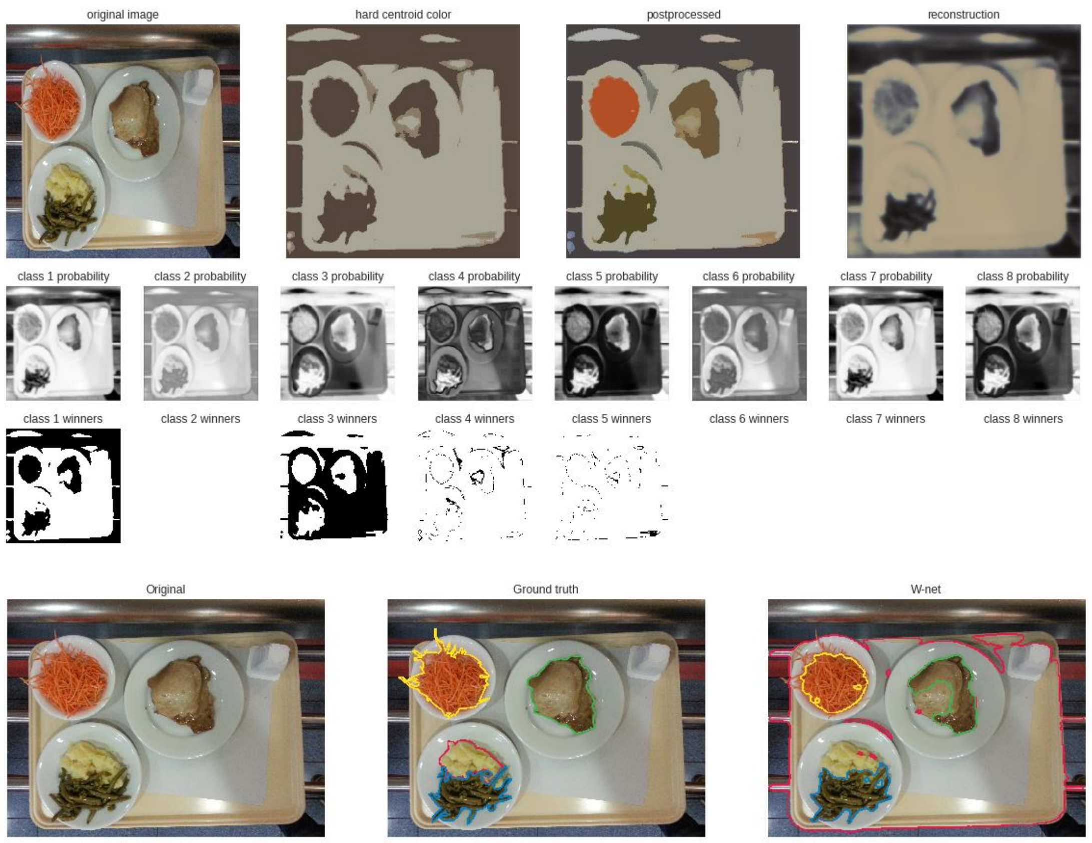


FIGURE 4.12: Results for the W-net 18-32

4.7.3 W-net 14-64

In this architecture, we decrease the number of modules from 18 to 14 and we make starting 3x3 convolutions which map the image into 64 feature channels (as in the W-net 18-64). This network has 12,256,134 parameters.

The mean IoU obtained for this architecture is **0.37**. When looking at the visualization, available in Figure 4.13, we can see that in this case we only obtain 2 classes for the hard segmentation, and the food is mainly contained in class 3. As happened before, we obtain good results after postprocessing for all the food except for the class "patate/puré", which in this case is not detected as a class, as we can see in the contour plot.

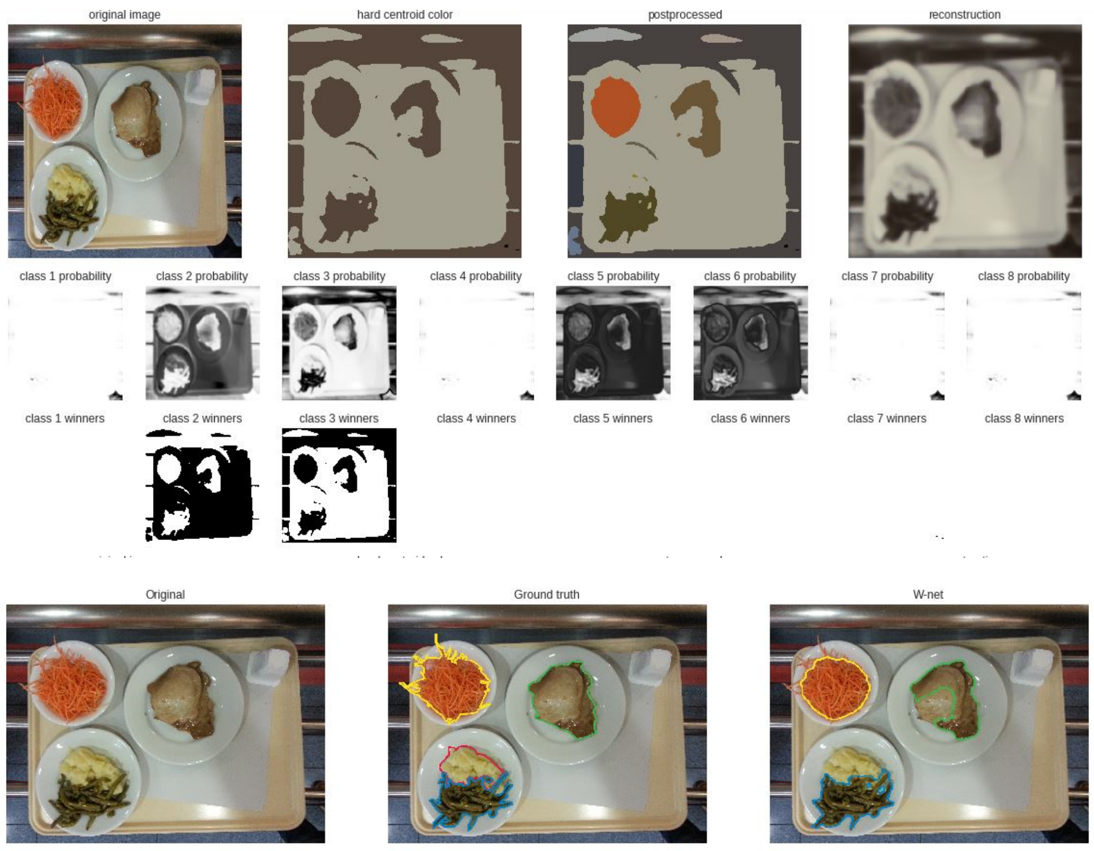


FIGURE 4.13: Results for the W-net 14-64

4.7.4 W-net 10-64

Finally, we decrease again the number of modules from 14 to 10 and start with 64 feature channels in the first module to obtain the W-net 10-64. The number of parameters in this model is 2,942,982.

The mean IoU for this architecture is **0.43**. Looking at the visualization, Figure 4.14, we can see that this segmentation looks a little better than the one obtained from the W-net 14-64. In this case, the class "cotoletta" gets fully recovered, when in the segmentation from the W-net 14-64, we only have some part of it. In this case, we obtain 3 classes, and the food is mainly contained in class 3. We can see that as happened before, in this case we also do not detect the "patate/puré" class, which again gets lost with the background as we can see in the contour plot.

To sum up, we have tested the different architectures, and we can see that we have the best results (see Table 4.2) using the W-net 18-32. If we look at the evolution of the sum of the segmentation and reconstruction losses in Figure 4.17, we can see that the loss from the W-net 18-32, decreases in a smoother way than the other architectures. In the end, we choose the W-net 18-32, which has 18 modules and starts with 32 feature channels instead of 64. We use this architecture with 0.2 dropout. From now on, we test with different optimizers and kernel initilalizers using this architecture.

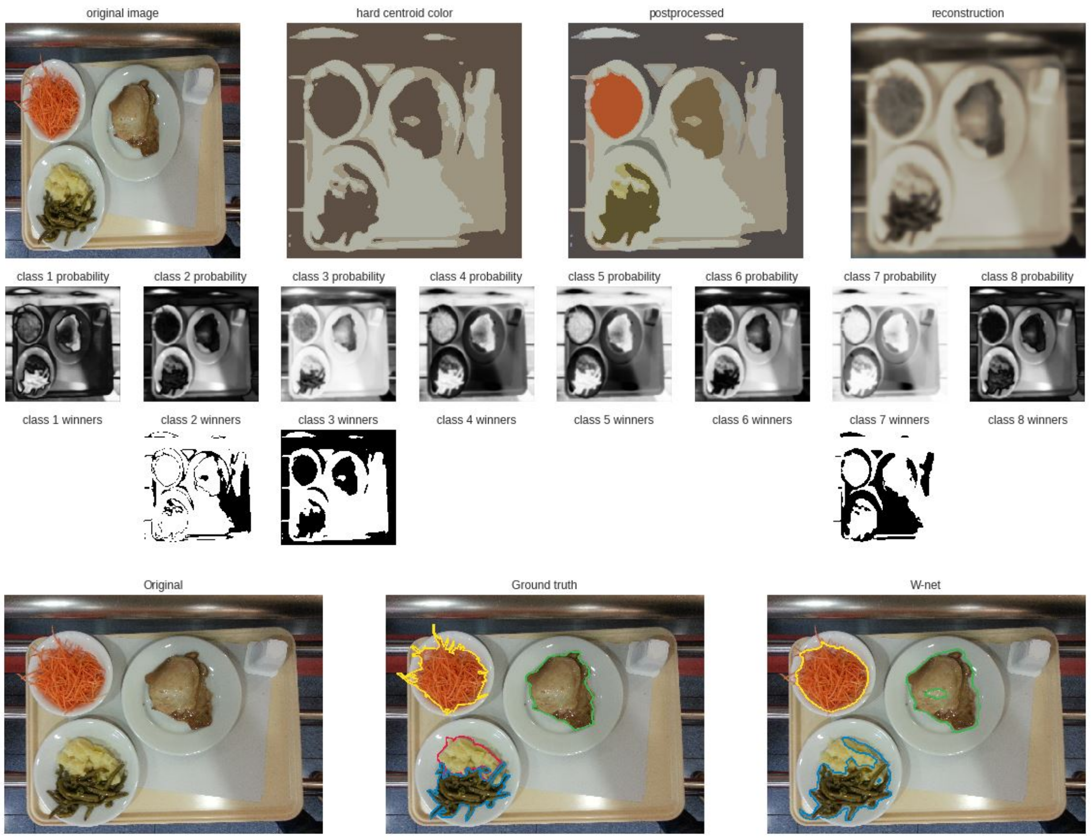


FIGURE 4.14: Results for the W-net 10-64

Architecture	IoU
W-net 18-64	0.29
W-net 18-32	**0.52**
W-net 14-64	0.37
W-net 10-64	0.43

TABLE 4.2: Results for the different W-net architectures

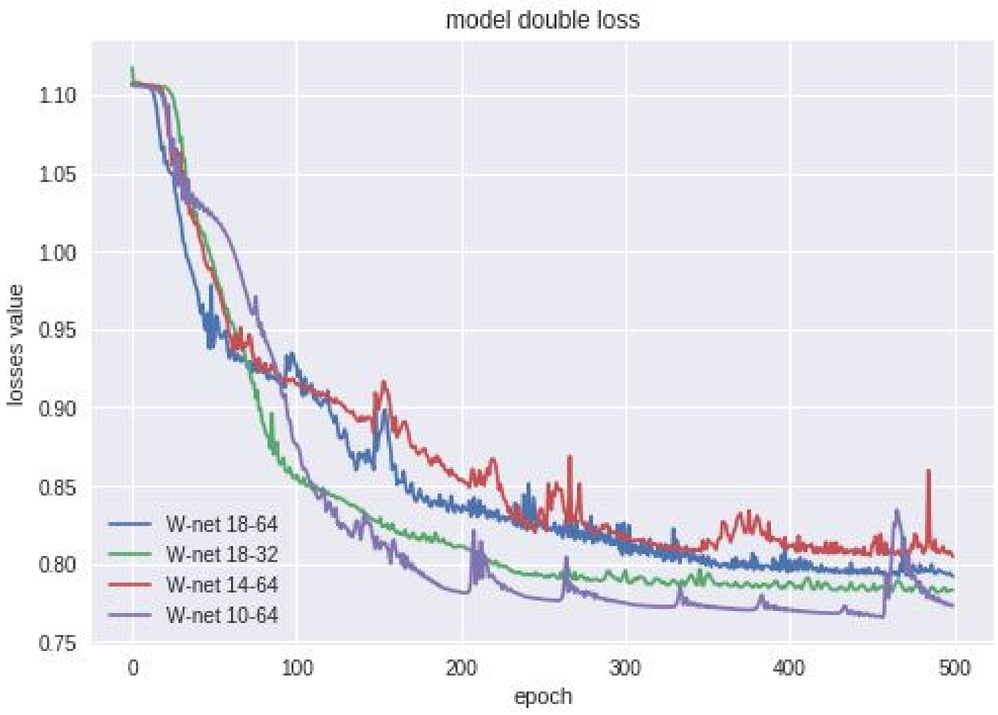


FIGURE 4.15: Evolution of the loss function for every architecture

4.8 Optimizers and Kernel Initializers

4.8.1 Optimizers

In order to obtain the best segmentation possible, we check different optimizers for the best architecture mentioned above. We start checking *Stochastic Gradient Descent* which is the optimizer used in (Xia and Kulis, 2017). We continue by testing *Adam*, *Adamax*, *Nadam*, *Adagrad* with different learning rates and decay.

For *Stochastic Gradient Descent*, we obtain an IoU of **0.25**, which is a much worse result than the one obtained for *Adam*, which gets an IoU of **0.52**.
If we compare the image visualizations obtained using both optimizers, see Figure 4.16 and Figure 4.12 (above), we see that using *SGD* we obtain regions which are not smooth, and this way the regions are not well defined as we can see in the contour plot. On the contrary, the results using *Adam* as an optimizer show smooth regions, from which it is easy to obtain the different classes. In addition, there is no reconstruction of the image.

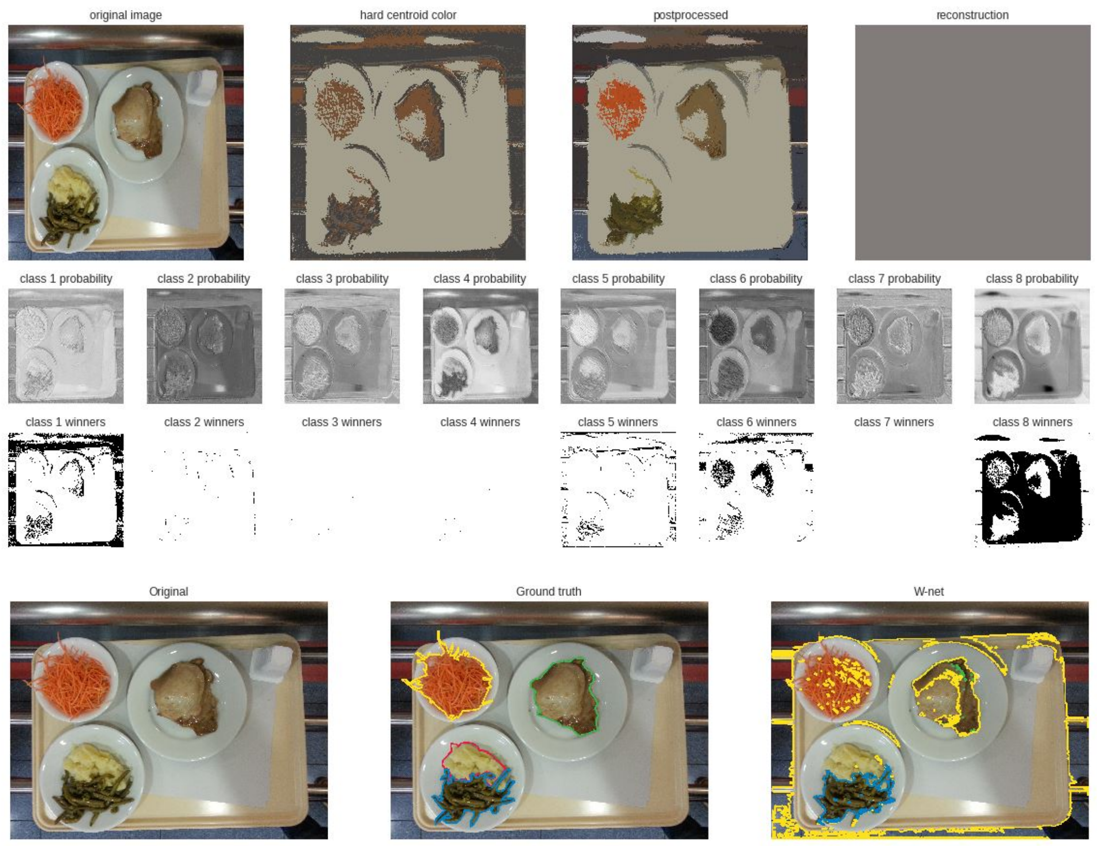


FIGURE 4.16: Results for the W-net 18-32 using SGD optimizer

If we compute the results for the other optimizers, we see that we obtain similar results than using *Adam*, although the best result is obtained using *Adam*. See Table 4.3 to see the mean IoU for every optimizer.
Looking at the evolution of the loss function (sum of segmentation and reconstruction loss), we can see that the *Adam* and *Adamax* losses decrease in a smooth way, opposite to *Adagrad* and *Nadam*, which have a lot of peaks, and the *SGD*, which does not seem to improve much. It is also important to mention that *Adamax* has also a high IoU value (0.48). However, it is not higher than *Adam*.

From this information, we conclude that the best architecture is the W-net 18-32 using *Adam* as optimizer.

Optimizer	IoU
`SGD`	0.25
`Adam`	**0.52**
`Adamax`	0.48
`Nadam`	0.20
`Adagrad`	0.19

TABLE 4.3: Results of the W-net 18-32 using different optimizers

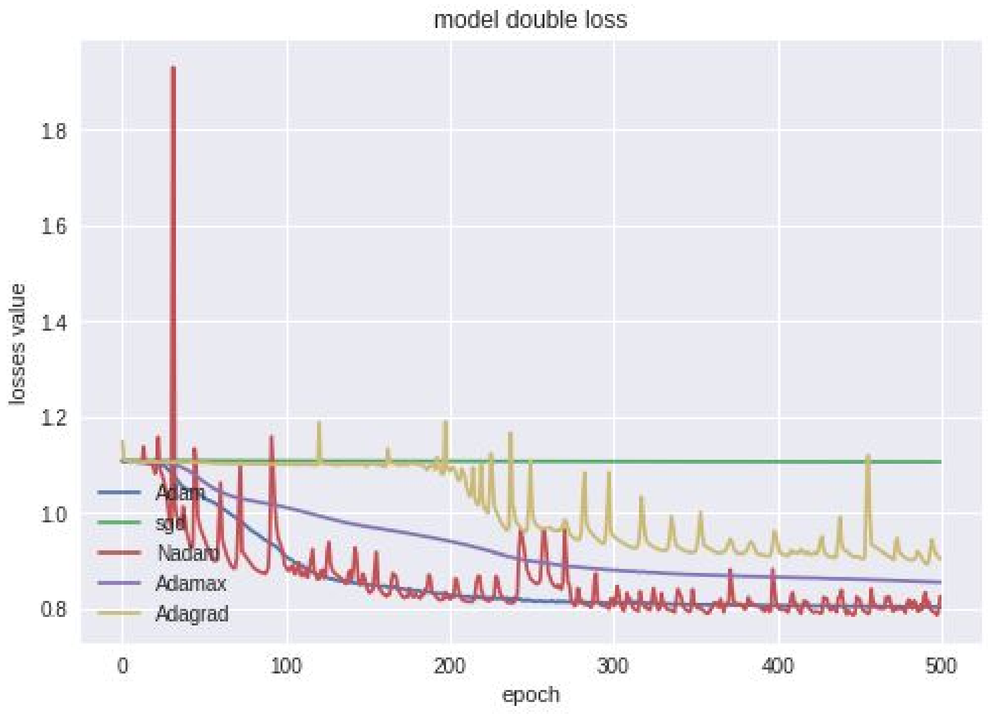


FIGURE 4.17: Evaluation of the loss function for W-net 18-32 with every optimizer

4.8.2 Kernel Initializers

When it comes to Kernel Initialization, the best choice is *He normal*. *He normal* draws samples from a truncated normal distribution centered on 0 with $stddev = \sqrt{\frac{2}{fan-in}}$, where $fan - in$ is the number of input units in the weight tensor and makes the model perform the best possible (He et al., 2015). *He Uniform*, *Uniform* and *Random Normal* with different standard deviation values are also tested, however the performance is lower.

He Uniform

Using *He Normal* as a kernel initializer in our best current architecture, we obtain good results after postprocessing which can be seen in Figure 4.18. The only class that can not be detected is the "patate/puré". The IoU value of this model is 0.44, which is a relatively high value, but still not higher than the 52 we obtain by using the *He Normal*.

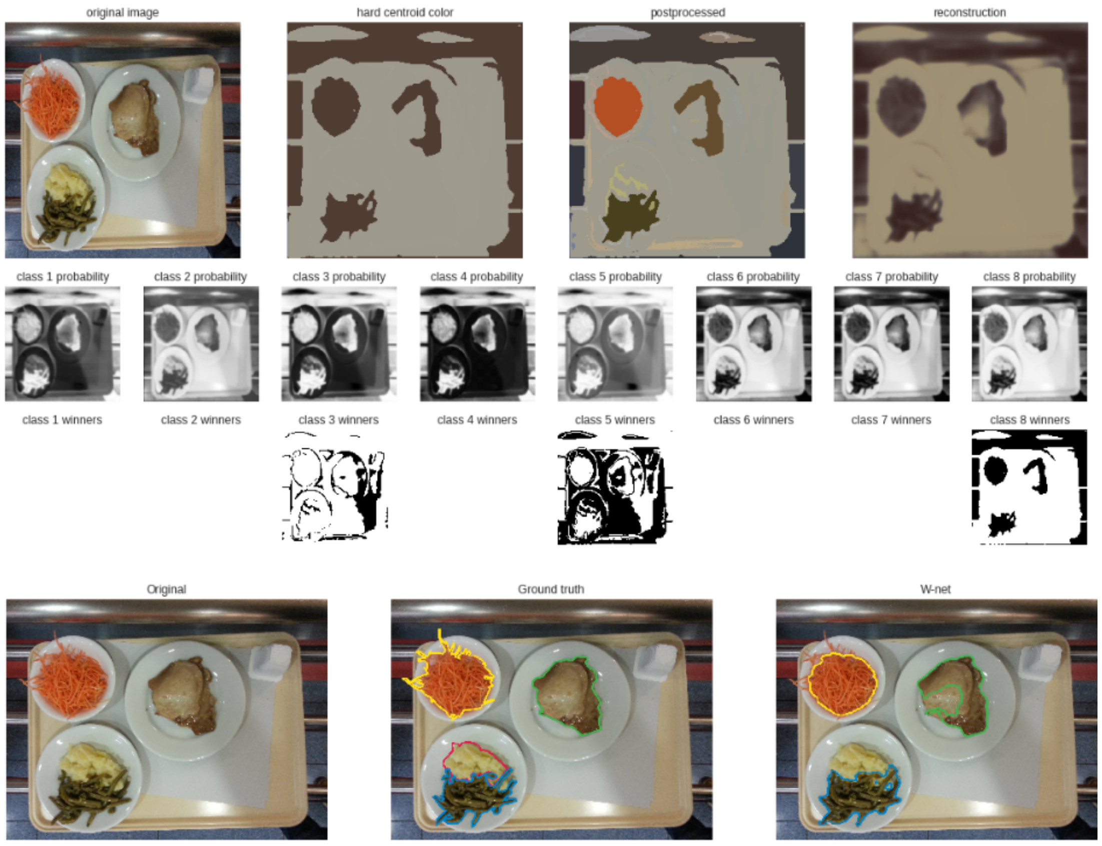


FIGURE 4.18: Results for the W-net with He Uniform Kernel initializer

Random Normal

Using *Random Normal* with mean 0 and 0.5, we obtain an IoU value of 0.14. This value is the worst value of segmentation we obtained in our test. In Figure 4.19, we can visualize the results of this model. Looking at the reconstruction, we can understand that this initialization is not the best choice for out model and even with many epochs the model does not reconstruct the image. At the same time the segmentation is not good even if the model detects the food. This can be definitely visualized in the contour where looking at the class "carote" the model creates small regions that make them not well defined.

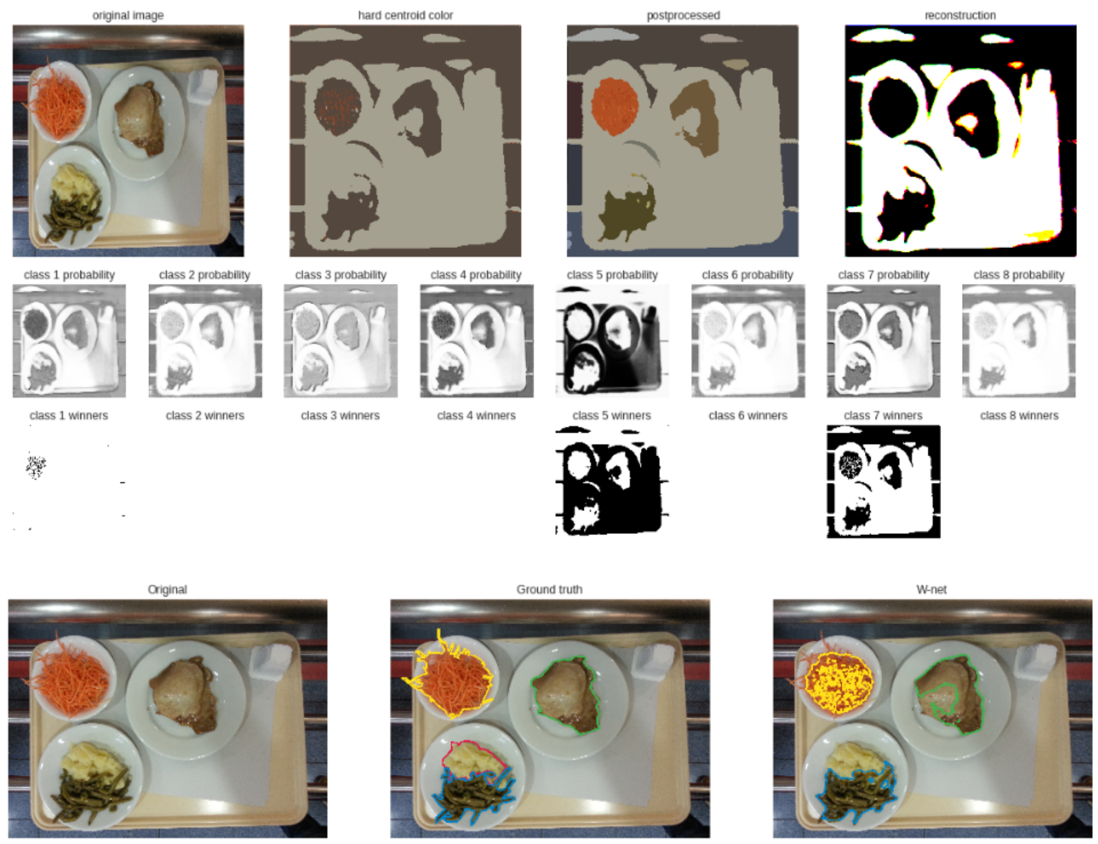


FIGURE 4.19: Results for the W-net with Random Normal Kernel initializer

4.8.3 W-net without Centroid Loss

Since we obtain the best results of the W-net using the *Centroid Loss* we will now compare these results with the results of the output of the final softmax layer in U_{Enc} without adding the *Centroid Loss*. In Figure 4.20 we can see that by not using the *Centroid Loss* we obtain a really good reconstruction however the segmentation is not satisfying. To be more precise, the regions are not smooth and consequently not well defined. The IoU of this segmentation equals to 0.20 which is much lower than 0.52 which is the IoU value of the same architecture adding the *Centroid Loss*. Also, looking at the contour we can also interpret that the segmentation is poor. For example, looking at the plate with the class "carote" we can see that there is no region created around it.

4.8.4 3D Visualization of pixel classification of our final model

Our most accurate model is the W-net 18-32 with *Adam* as an optimizer and *He Normal* as a kernel initializer. The results of this model, as it has been mentioned before, obtain a mean IoU of **0.52**. In Figure 4.21 we can visualize the performance of this model on the same image we are testing all the unsupervised W-net models into a 3D plot. To be more precise, we define our three dimensional space as x=R, y=G, z=B. The colour of each point for the first plot is the colour of the pixels in the original image, for the second the colour of the centroid where this pixel belongs and for the third plot the colour of the pixels obtained after postprocessing.

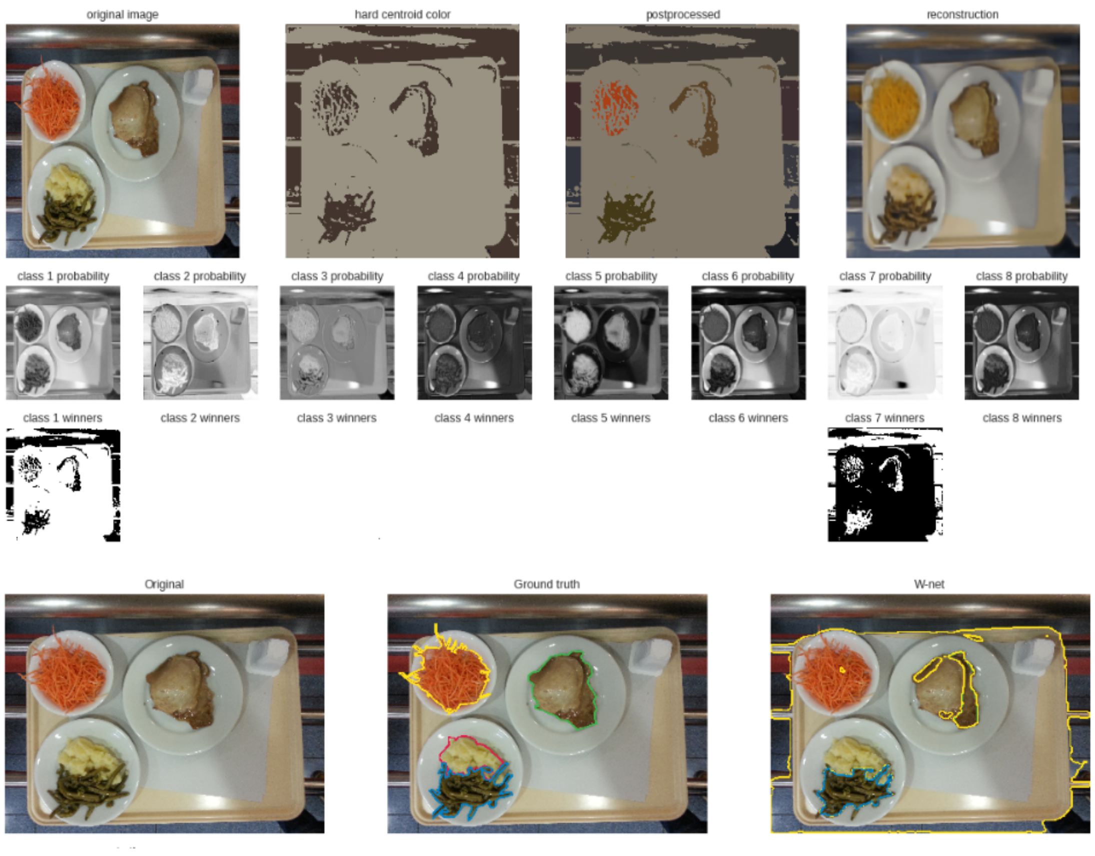


FIGURE 4.20: Results for the W-net without Centroid Loss

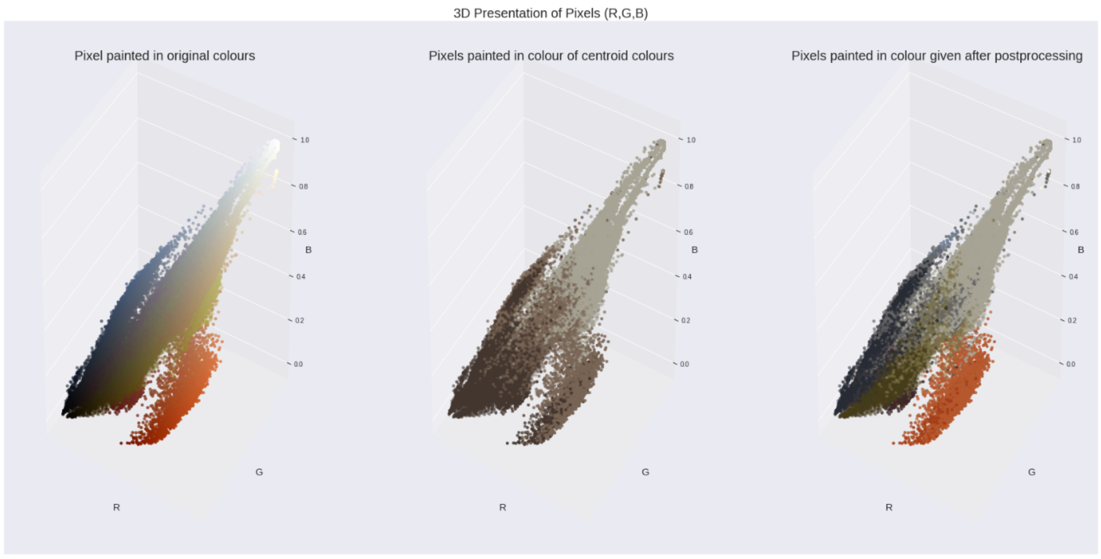


FIGURE 4.21: 3D pixel Visualization

4.9 Supervised W-net

Once we have decided which is the best architecture for the W-net, we try to test the architecture using the ground truth for the segmentation loss. The idea behind this is to train the supervised W-net and then train images unseen for the training or from different datasets with the *Centroid Loss* and see if our network is capable of predicting new classes of food.

First of all we train the W-net 18-32 with the ground truth, and we obtain an IoU of **0.65**. We compute the visualization with this network, see Figure 4.22. As expected, we can see that the segmented image gets rid of the background in this case. We also observe that the algorithm cannot reconstruct the segmented parts detected. In the contour plot we can see that the segmentation is quite good, although some classes in the same plate have been blended together.

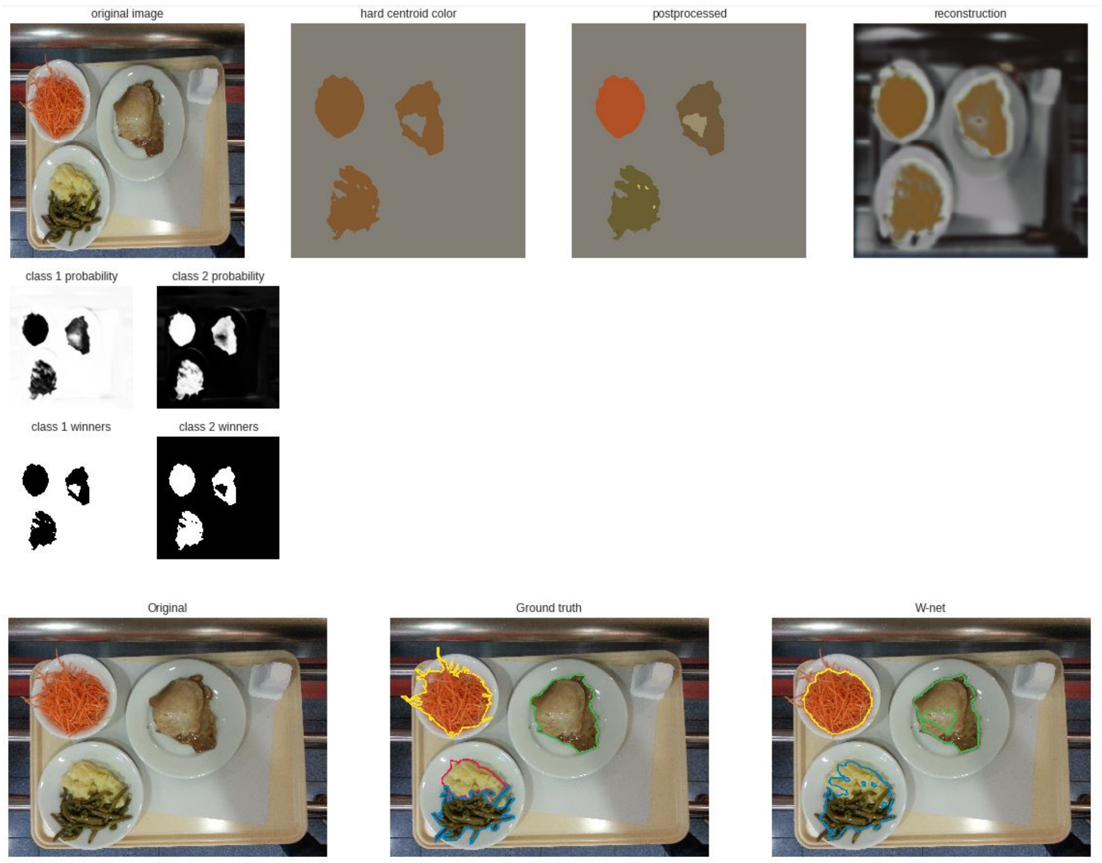


FIGURE 4.22: Results for the supervised W-net 18-32

Once we have the supervised W-net, we try to train images which have classes unseen in training using the *Centroid Loss*. In Figure 4.23 we can see that in this case we do not get a good segmentation. In fact the mean IoU for this results is **0.06**. It is evident in Figure 4.23 that the network can not obtain unseen classes, as it only predicts one class which is in the training set. We can also see that the segmentation that is found is not very good. In the end we discard this method due to the poor results obtained, although we think it is a very interesting path to study in future work for this network.

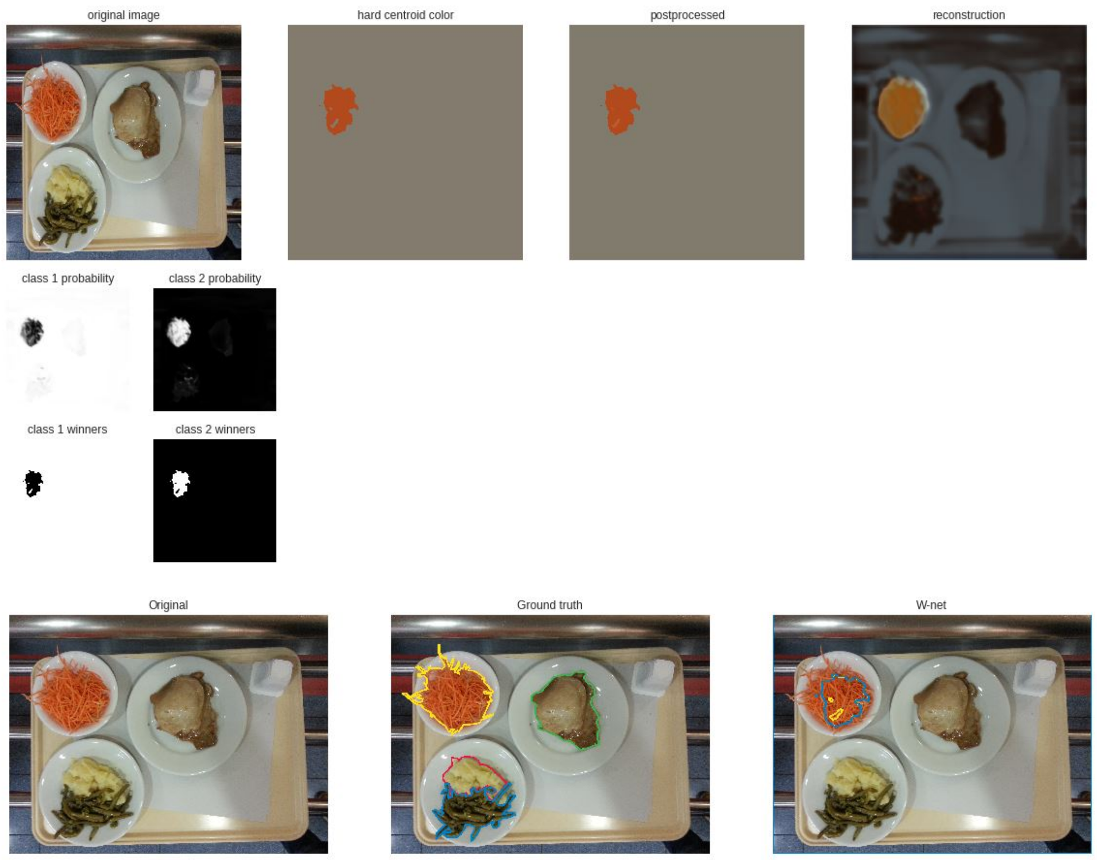


FIGURE 4.23: Results for the W-net 18-32 with weights from the supervised W-net

4.10 Result comparison

In order to give a good sense of the main difference with the main architectures developed in the project, we put all the results together. In Table 4.4 we can see the comparison with the main architectures developed. As we can see, for the supervised networks, the binary U-net gives very high results. As for the unsupervised architectures, we can see that the best result we obtain is for the W-net 18-34. It doesn't have a very high IoU value, but for unsupervised segmentation, the results are quite remarkable. Also they are slightly higher than the embedding results, so using convolutions and reconstruction helps.
We can also see a comparison of the visualization for these models in Figure 4.24, Figure 4.25 and Figure 4.26.

Architecture	IoU
Binary U-net	**0.87**
Multiclass U-net	0.65
Supervised W-net	0.65
Embedding	0.45
W-net 18-34	**0.52**
Crossdomain W-net	0.06

TABLE 4.4: Results for the different networks reviewed in the project

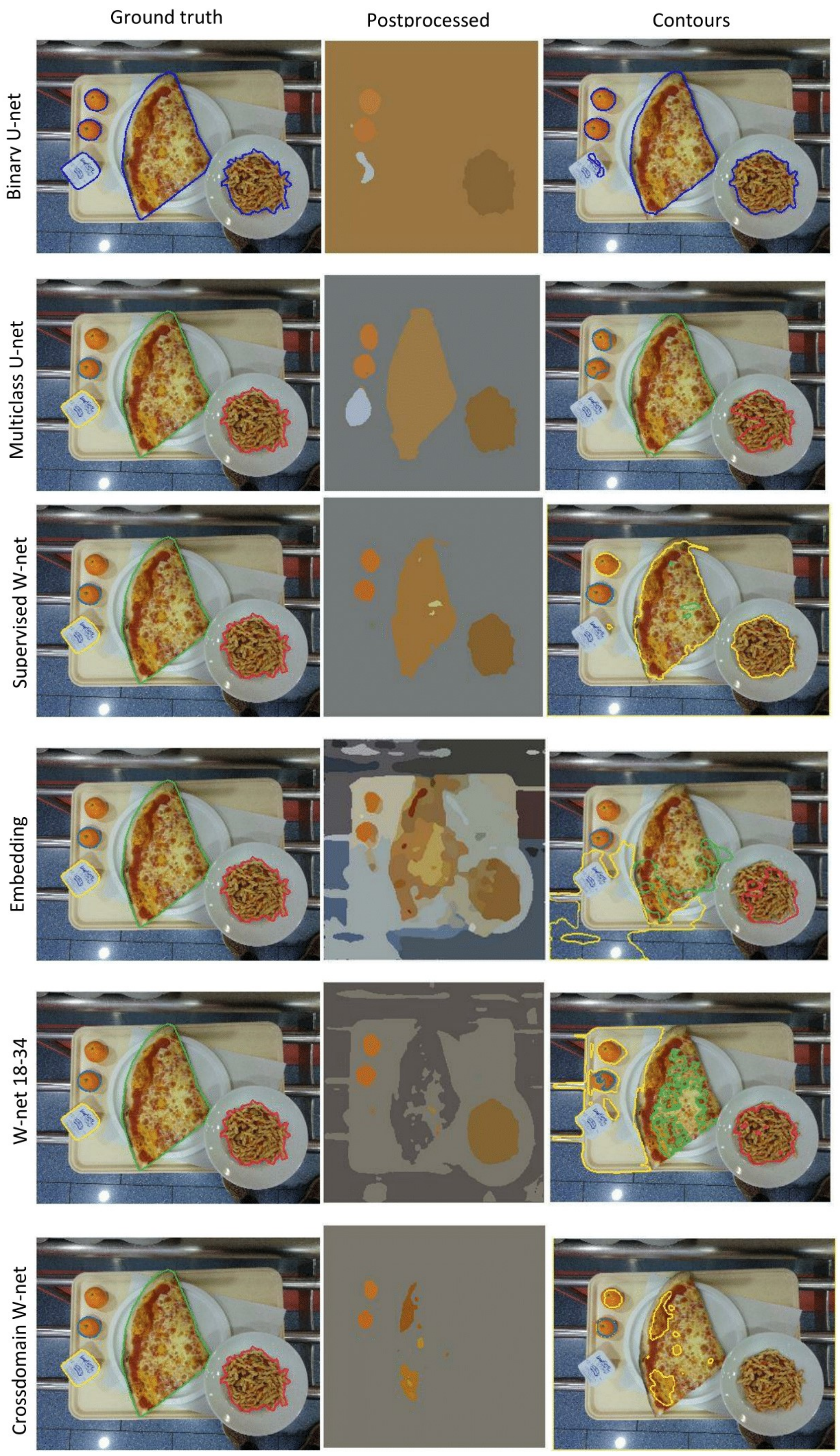


FIGURE 4.24: Comparison of the Binary U-net, Multiclass U-net, supervised W-net, embedding with the Centroid loss, W-net 18-34 and crossdomain W-net

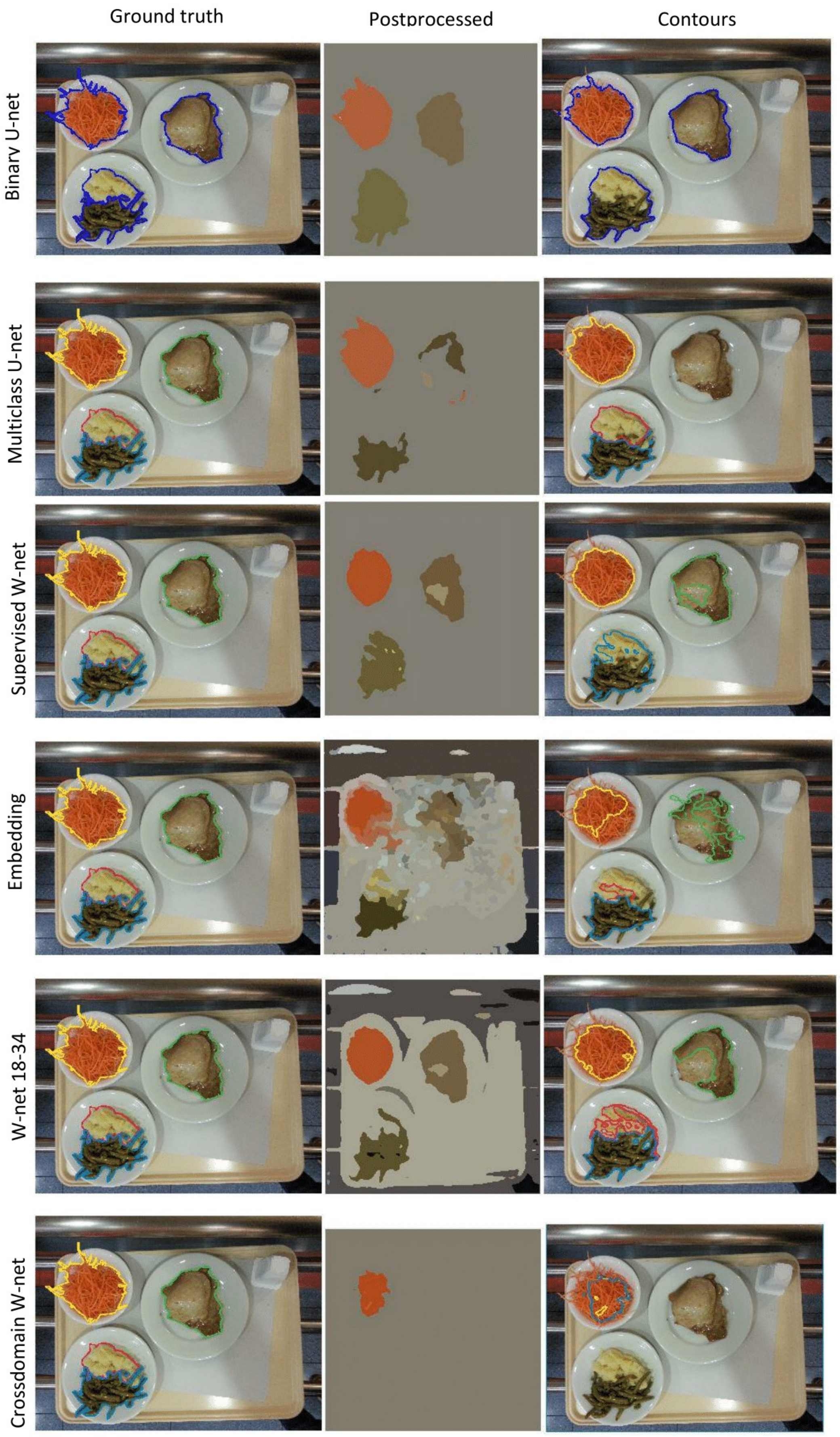


FIGURE 4.25: Comparison of the Binary U-net, Multiclass U-net, supervised W-net, embedding with the Centroid Loss, W-net 18-34 and crossdomain W-net

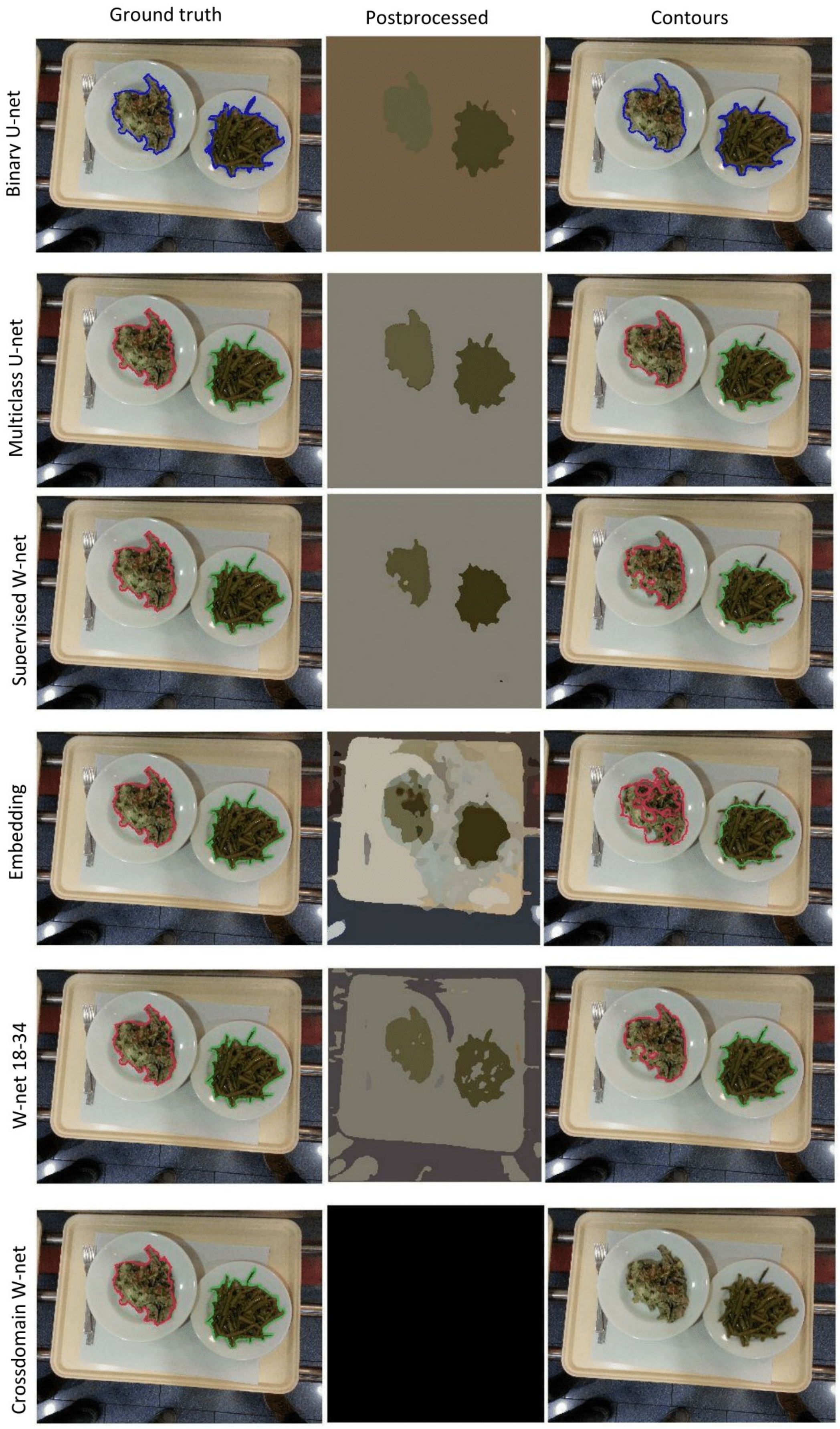


FIGURE 4.26: Comparison of the Binary U-net, Multiclass U-net, supervised W-net, embedding with the Centroid Loss, W-net 18-34 and crossdomain W-net

Chapter 5

Conclusions

In this project, we have undertaken the task of studying the possibility of successfully using unsupervised Convolutional Neural Networks for segmentation applied to food analysis.

First of all, we have reviewed the most relevant works on segmentation, focusing on unsupervised segmentation and convolutional neural networks. Then we have given a detailed review of the two networks that we focus most on this project, the W-net and the U-net. We have given the details of the architectures as well as the results obtained for these networks regarding segmentation.
Following this, we have provided a full explanation of the "soft" normalized cut loss function, used in the W-net for minimizing the error of the segmentation. As this loss function has revealed infeasible to compute for reasonably sized images, we have proposed an alternative loss function for the segmentation in the W-net, which we have called *Centroid Loss*. It has proven computationally viable while preserving the essence of the original, and with relatively good results.

Regarding the experimental part of the project, we have used the UNIMIB2016 Food Database in order to compute results for the developed architectures. We have adapted the U-net architecture to fit our dataset and tested it with binary ground truth and multiclass ground truth with 73 classes. Both architectures have given good results.
We have developed the W-net from scratch with our *Centroid Loss* function and have tested it with different number of modules, feature channels, optimizers and kernel initializations. After making many experiments we have concluded that the best architecture for the W-net and for our dataset is the W-net 18-34, which has 38 modules just as the original W-net, but with 34 features channels in the first convolution. We have also tried the W-net 18-34 with the supervised loss, minimizing the difference with the ground truth and we have obtained similar results as for the multiclass U-net. Finally, we have tried to train the W-net 18-34 with the *Centroid Loss* initializing it with the weights from the supervised W-net in order to be able to predict classes not contained in the ground truth, but we have not obtained good results on this approach.

To sum up, we have been able to develop an unsupervised segmentation network using a new loss function that has different properties from the point of view of efficiency and compactness, after finding that the N-cut loss is not efficient or easy to apply to the network. We have also applied a much simpler postprocessing than the one used on the W-net original paper, and we have obtained good results for segmentation. Finally, we have been able to successfully apply our network to food images, which can help in the process of food analysis.

5.0.1 Limitations and difficulties

We have had to overcome the following challenges:

- N-cut loss: its implementation took surprisingly long to compute in tensorflow even for small images, which brings us to the next point
- Limited resources: in the absence of a server we have resorted to Google Colaboratory, a free service for running notebooks inside Google Drive with 12h of uninterrupted execution with available Tesla K80 GPU and 23Gb of RAM. This service is great, but suffers from spontaneous runtime death and additional difficulty to store data.
- Complex images: unsupervised semantic segmentation should not be intended for images with complex backgrounds, which already contain semantic content apart from our target, the food.
- No implementation was given in the W-net paper, and authors did not answer any message for clarification.

5.0.2 Next steps

Next steps are to explore:

- Transfer learning: We were unsuccessful to apply transfer learning, for our case, using the weights obtained from training the supervised W-net as initialization for the unsupervised W-net. The model evolved into a flat segmentation. We should explore why the semantic-optimal solution is not a stable minimum with respect to our unsupervised loss function. Is this to be expected? Could we fix it with a better architecture or a more refined loss function?
- More postprocessing: We kept the postprocessing very simple, just splitting classes into connected components. This is good because it doesn't introduce much bias about how solutions should look like, but there's still work to do on other common strategies, such as Conditional Random Fields.
- Testing on other datasets: it would be nice to see how it performs on other food images (or more diverse images).

Appendix A

Contribution of each team member

In this appendix we give a detailed summary of the work realized by each team member.

- Montserrat Brufau Vidal
  1. Preprocessing of the UNIMIB2016 dataset with Matlab.
  2. Development of the U-net architecture and tests with the UNIMIB2016 dataset.
  3. Development of the supervised W-net and tests with the UNIMIB2016 dataset.
  4. Tests for the crossdomain problem.
  5. IoU metric implementation.
  6. Contour visualization.
  7. Project report.
- Àlex Ferrer Campo
  1. Soft N-cut loss
  2. Optimizations on the Soft N-cut loss implementation: sum vectorization
  3. Design of the Centroid Loss and edge-penalty loss
  4. Testing them in the embedding network
  5. Mean color visualizations
  6. Postprocessing
  7. Project report
- Markos Gavalas
  1. N-cut loss
  2. W-net architecture development and plugging losses
  3. W-net tests with the N-cut loss
  4. W-net tests with the Centroid loss
  5. Results for the Unsupervised W-net
  6. 3D Pixel Visualization
  7. Project report

Bibliography

AA, Aly, Deris SB, and Zaki N (2011). "Research review for digital image segmentation techniques". In: pp. 1302–1317. URL: http://airccse.org/journal/jcsit/1011csit09.pdf.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2015). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *CoRR* abs/1511.00561. arXiv: 1511.00561. URL: http://arxiv.org/abs/1511.00561.

Chaurasia, Abhishek and Eugenio Culurciello (2017). "LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation". In: *CoRR* abs/1707.03718. arXiv: 1707.03718. URL: http://arxiv.org/abs/1707.03718.

Chen, Hsin-Chen et al. (2015). "Saliency-aware food image segmentation for personal dietary assessment using a wearable computer". In: *Measurement Science and Technology* 26.2, p. 025702. URL: http://stacks.iop.org/0957-0233/26/i=2/a=025702.

Ciocca, Gianluigi, Paolo Napoletano, and Raimondo Schettini (2017). "Food recognition: a new dataset, experiments and results". In: *IEEE Journal of Biomedical and Health Informatics* 21.3, pp. 588–598. DOI: 10.1109/JBHI.2016.2636441.

Comaniciu, Dorin and Peter Meer (2002). "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, pp. 603–619.

Cour, Timothee, Florence Benezit, and Jianbo Shi (2005). "Spectral Segmentation with Multiscale Graph Decomposition". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*. CVPR '05. Washington, DC, USA: IEEE Computer Society, pp. 1124–1131. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.332. URL: http://dx.doi.org/10.1109/CVPR.2005.332.

D, Weinland, Ronfard R, and Boyer E (2011). "A survey of vision-based methods for action representation, segmentation and recognition". In: *Computer Vision and Image Understanding*, 224–241. ISSN: 1945-7871. DOI: 10.1016/j.cviu.2010.10.002. URL: https://www.sciencedirect.com/science/article/pii/S1077314210002171.

Dai, Jifeng, Kaiming He, and Jian Sun (2014). "Convolutional Feature Masking for Joint Object and Stuff Segmentation". In: *CoRR* abs/1412.1283. arXiv: 1412.1283. URL: http://arxiv.org/abs/1412.1283.

DD, Patil and Deore SG (2013). "Medical image segmentation: a review". In: 22–27. ISSN: 2320–088X. URL: https://pdfs.semanticscholar.org/56f7/0bafc5002ae723d6ee20ce9087f6eff7f81e.pdf.

DE, Ilea and Whelan PF (2011). "Image segmentation based on the integration of colour-texture descriptors—a review". In: 44, pp. 2479–2501. DOI: 10.1016/j.patcog.2011.03.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320311000902.

Farabet, Clement et al. (2012). "Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images". In: pp. 1915 –1929. ISSN: 0162-8828. DOI: 10.1109/

TPAMI.2012.231. URL: http://yann.lecun.com/exdb/publis/pdf/farabet-pami-13.pdf.

Felzenszwalb et al. (2011). “Image Segmentation Using Local Variation and Edge-Weighted Centroidal Voronoi Tessellations”. In: pp. 3242 –3256. DOI: 10.1109/TIP.2011.2150237. URL: https://pdfs.semanticscholar.org/5f73/6c61cd7b3d1019241de0de1bea7e9bae885f.pdf.

Felzenszwalb, Pedro F. et al. (2009). “Object Detection with Discriminatively Trained Part-Based Models”. In: pp. 1627 –1645. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2009.167. URL: https://ieeexplore.ieee.org/document/5255236.

Geman, Stuart (1984). “Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images”. In: 721–741. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1984.4767596. URL: https://www.sciencedirect.com/science/article/pii/S0031320311000902.

Guo, Yanming et al. (2018). “A review of semantic segmentation using deep neural networks”. In: *International Journal of Multimedia Information Retrieval* 7.2, pp. 87–93. ISSN: 2192-662X. DOI: 10.1007/s13735-017-0141-z. URL: https://doi.org/10.1007/s13735-017-0141-z.

Hariharan, Bharath et al. (2014a). “Hypercolumns for Object Segmentation and Fine-grained Localization”. In: *CoRR* abs/1411.5752. arXiv: 1411.5752. URL: http://arxiv.org/abs/1411.5752.

Hariharan, Bharath et al. (2014b). “Simultaneous Detection and Segmentation”. In: *CoRR* abs/1407.1808. arXiv: 1407.1808. URL: http://arxiv.org/abs/1407.1808.

He, Kaiming et al. (2015). “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *CoRR* abs/1502.01852. arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852.

He, Ye et al. (2013). “Food image analysis: Segmentation, identification and weight estimation”. In: ISSN: 1945-7871. DOI: 10.1109/ICME.2013.6607548. URL: https://ieeexplore.ieee.org/document/6607548.

Krähenbühl, Philipp and Vladlen Koltun (2012). “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials”. In: *CoRR* abs/1210.5644. arXiv: 1210.5644. URL: http://arxiv.org/abs/1210.5644.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: pp. 1097–1105. URL: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

LeCun, Y. et al. (1989). “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *CoRR* 1, pp. 541 –551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. URL: https://ieeexplore.ieee.org/document/6795724.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2014). “Fully Convolutional Networks for Semantic Segmentation”. In: *CoRR* abs/1411.4038. arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

M, Sonka, Hlavac V, and Boyle R (2014). “Image processing, analysis, and machine vision”. In: DOI: 10.1109/ICME.2013.6607548. URL: https://www.springer.com/la/book/9780412455704.

Matsuda, Yuji, Hajime Hoashi, and Keiji Yanai (2012). “Recognition of Multiple-Food Images by Detecting Candidate Regions”. In: 1554–1564. DOI: 10.1109/ICME.2012.157. URL: https://www.semanticscholar.org/paper/Recognition-of-Multiple-Food-Images-by-Detecting-Matsuda-Hoashi/1631058c2423017735db20431cdafadffee6e738?tab=abstract.

Mostajabi, Mohammadreza, Payman Yadollahpour, and Gregory Shakhnarovich (2014). "Feedforward semantic segmentation with zoom-out features". In: *CoRR* abs/1412.0774. arXiv: 1412.0774. URL: http://arxiv.org/abs/1412.0774.

MW, Khan (2014). "A survey: image segmentation techniques". In: URL: http://ijfcc.org/papers/274-B317.pdf.

Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han (2015). "Learning Deconvolution Network for Semantic Segmentation". In: *CoRR* abs/1505.04366. arXiv: 1505.04366. URL: http://arxiv.org/abs/1505.04366.

P, Arbelaez et al. (2010). "Contour Detection and Hierarchical Image Segmentation". In: pp. 898 –916. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.161. URL: https://ieeexplore.ieee.org/document/5557884.

P, Yu, Qin AK, and Clausi DA (2011). "Unsupervised Polarimetric SAR Image Segmentation and Classification Using Region Growing With Edge Penalty". In: pp. 1302–1317. ISSN: 0196-2892. DOI: 10.1109/TGRS.2011.2164085. URL: https://ieeexplore.ieee.org/document/6020785.

Paszke, Adam et al. (2016). "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation". In: *CoRR* abs/1606.02147. arXiv: 1606.02147. URL: http://arxiv.org/abs/1606.02147.

R, Muthukrishnan and Radha M (2011). "Edge detection techniques for image segmentation". In: URL: http://airccse.org/journal/jcsit/1211csit20.pdf.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597. arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

Shi, Jianbo and Jitendra Malik (2000). "Normalized Cuts and Image Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.8, pp. 888–905. ISSN: 0162-8828. DOI: 10.1109/34.868688. URL: https://doi.org/10.1109/34.868688.

Shimoda, Wataru and Keiji Yanai (2015). "CNN-Based Food Image Segmentation Without Pixel-Wise Annotation". In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*. Ed. by Vittorio Murino et al. Cham: Springer International Publishing, pp. 449–457. ISBN: 978-3-319-23222-5.

Simonyan, Karen and Andrew Zisserman (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556. arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

SR, Vantaram and Saber E (2012). "Survey of contemporary trends in color image segmentation". In: pp. 1302–1317. ISSN: 0196-2892. DOI: 10.1117/1.JEI.21.4.040901. URL: https://www.spiedigitallibrary.org/journals/Journal-of-Electronic-Imaging/volume-21/issue-4/040901/Survey-of-contemporary-trends-in-color-image-segmentation/10.1117/1.JEI.21.4.040901.full?SSO=1.

T, Zuva et al. (2011). "Image segmentation, available techniques, developments and open issues". In: *Journal of Signal and Image Processing*, pp. 71–75. URL: https://www.researchgate.net/publication/264854010_Image_Segmentation_Available_Techniques_Developments_and_Open_Issues.

Thoma, Martin (2016). "A Survey of Semantic Segmentation". In: *CoRR* abs/1602.06541. arXiv: 1602.06541. URL: http://arxiv.org/abs/1602.06541.

Xia, Xide and Brian Kulis (2017). "W-Net: A Deep Model for Fully Unsupervised Image Segmentation". In: *CoRR* abs/1711.08506. arXiv: 1711.08506. URL: http://arxiv.org/abs/1711.08506.

Zhang, Y., M. Brady, and S. Smith (2001). "Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm". In: *IEEE Transactions on Medical Imaging* 20.1, pp. 45 –57. ISSN: 0162-8828. DOI: 10.1109/42.906424. URL: https://ieeexplore.ieee.org/document/906424.

Zhao, Bo et al. (2017). "A survey on deep learning-based fine-grained object classification and semantic segmentation". In: *International Journal of Automation and Computing* 14.2, pp. 119–135. ISSN: 1751-8520. DOI: 10.1007/s11633-017-1053-3. URL: https://doi.org/10.1007/s11633-017-1053-3.

Zheng, Shuai et al. (2015). "Conditional Random Fields as Recurrent Neural Networks". In: *CoRR* abs/1502.03240. arXiv: 1502.03240. URL: http://arxiv.org/abs/1502.03240.