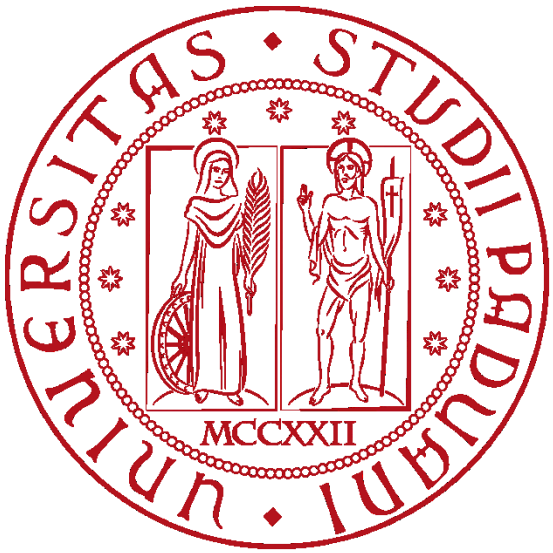
An empirical study on ensemble of segmentation approaches



University of Padova - Department of Information Engineering

BSc in Computer Engineering

Academic Year 2021-2022

Supervisor

Prof. Loris Nanni

Author

Alberto Formaggio

July 2022

To my family, who always supported me and
gave me the opportunity to pursue higher education.

Abstract (English version)

Recognizing objects in images requires complex skills that involve knowledge about the context and the ability to identify the borders of the objects. In computer vision, this task is called semantic segmentation and it pertains to the classification of each pixel in an image. The task is of main importance in many real-life scenarios: in autonomous vehicles, it allows the identification of objects surrounding the vehicle; in medical diagnosis, it improves the ability of early detecting dangerous pathologies and thus to mitigate the risk of serious consequences. In this work, we propose a new ensemble method able to solve the semantic segmentation task. The model is based on convolutional neural networks (CNNs) and transformers. An ensemble uses many different models whose predictions are aggregated to form the output of the ensemble system. The performance and quality of the ensemble prediction are strongly connected with some factors, one of the most important is the diversity among individual models. In our approach, this is enforced by adopting different loss functions and testing different data augmentation. We developed the proposed method by combining DeepLabV3+, HarDNet-MSEG, and Pyramid Vision Transformers. The developed solution was then assessed through an extensive empirical evaluation in five different scenarios: polyp detection, skin detection, leukocytes recognition, environmental microorganism detection, and butterfly recognition. The model provides state-of-the-art results. All resources will be available online at https://github.com/AlbertoFormaggio1/Ensemble-Of-Segmentation.

Sommario (versione Italiana)

Riconoscere oggetti all'interno delle immagini richiede delle abilità complesse che richiedono una conoscenza del contesto e la capacità di identificare i bordi degli oggetti stessi. Nel campo della computer vision, questo compito è chiamato segmentazione semantica e riguarda la classificazione di ogni pixel all'interno di un'immagine. Tale compito è di primaria importanza in molti scenari reali: nei veicoli autonomi, dove permette l'identificazione degli oggetti che circondano il veicolo, o nella diagnosi medica, in cui migliora la capacità di identificare patologie pericolose e quindi mitigare il rischio di serie conseguenze. In questo studio, proponiamo un nuovo modello per un multiclassificatore in grado di risolvere il compito di segmentazione semantica. Il modello si basa su reti neurali convoluzionali (CNN) e transformers. Un multiclassificatore usa diversi modelli le cui stime vengono aggregate così da ottenere l'output del sistema di multiclassificazione. Le prestazioni e la qualità delle previsioni dell'ensemble sono fortemente connessi ad alcuni fattori, tra cui il più importante è la diversità tra i singoli modelli. Nell'approccio qui proposto, abbiamo ottenuto questo risultato adottando diverse loss functions e testando con diversi metodi di data augmentation. Abbiamo sviluppato questo metodo combinando DeepLabV3+, HarDNet-MSEG e dei Pyramid Vision Transformers (PVT). La soluzione qui sviluppata è stata poi esaminata mediante un'ampia valutazione empirica in 5 diversi scenari: rilevamento di polipi, rilevamento della pelle, riconoscimento di leucociti, rilevamento di microorganismi e riconoscimento di farfalle. Il modello fornisce dei risultati che sono allo stato dell'arte. Tutte le risorse sono disponibili online all'indirizzo https://github.com/AlbertoFormaggio1/Ensemble-Of-Segmentation.

Contents

Chapter 1
Introduction

Being able to recognize objects in images has been for a long time a prerogative of human beings. It has taken over 14 years to reach the level of an untrained human in the challenge of Imagenet. Things become more complex when the task requires not only to recognize the object in an image but also to identify its boundaries. This task is called semantic segmentation and in machine learning this entails the classification of each pixel in an image. Due to the improvements of performance related to the adoption of machine learning models, this task is applied to many real-life scenarios [1,2]: in clinical practice, it can be used to identify polyps, similarly, in skin and blood analysis the identification of objects may help to visually bound the presence of different types of diseases. In addition, it can be used in autonomous vehicles, to identify objects surrounding the vehicle, in classification of environmental microorganisms, and in many other contexts.

The standard approach is to train a system composed of two modules: an encoder, and a decoder. The first module learns a low-dimensional representation of the input that describes semantics in the image. The second module learns to build the original input based on this low-dimensional feature vector. This has been the approach adopted by U-Net [3], one of the first systems developed for semantic segmentation.

Autoencoders [4] were also employed to resolve the task due to their ability to learn the semantics of low-level representations of an image through the encoder module as well as the ability to re-construct the original input from this reduced representation. Autoencoders performance and results are the reasons why many researchers and practitioners from the computer vision area have adopted them.

The performance of autoencoders, as well as the ones of other classifier technologies, are strongly affected by architecture configurations, and other configurations often referred to as

hyper-parameters tuning. That consists in finding the best values of some attributes of the model. This is a context-specific task that requires domain knowledge as well as expertise with the adopted machine learning techniques, resulting in big efforts and time consumption. The well-known no-free lunch theorem for machine learning highlights that a single model that works well on all the datasets cannot exist. Based on this evidence, another approach consists in adopting sets of classifiers, often shallow or weak, whose predictions are aggregated to form the output of the system. These frameworks are called ensemble methods. Roughly speaking, each classifier can be viewed as a voter in an election who expresses its preference on a set of possible alternatives, then the one that gets the majority of the votes is the chosen one for that election. In an ensemble, individual classifiers are trained on the same dataset, in such a way each model should generalise differently in the training space. Ensembles provide state-of-the-art results in many domains, but it is important to ensure some properties. One of them is to enforce some kind of diversity in the set of classifiers.

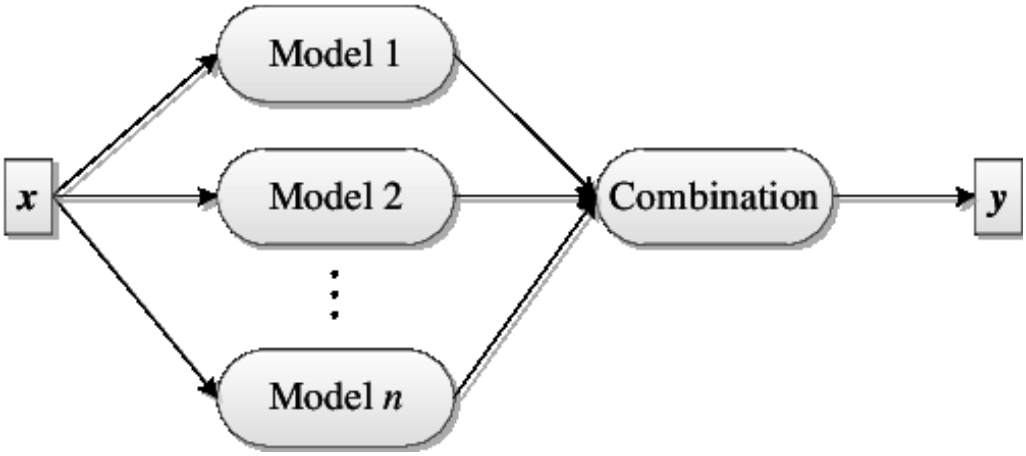


Figure 1. The common ensemble architecture

In this work, we propose a novel ensemble method for semantic segmentation. Our model is based on convolutional neural networks (CNNs) and transformers. Diversity among individual classifiers is enforced by adopting different loss functions and testing different data augmentation.

The model has been developed by combining DeepLabV3+ [5], HarDNet-MSEG [6], and Pyramid Vision Transformers[7]. We tested our proposal in five real-world scenarios: polyp detection, skin detection, leukocytes recognition, environmental microorganism detection and butterfly recognition. The developed solution was then assessed through an extensive empirical evaluation that compares our proposal with state-of-the-art solutions highlighting promising results often better than the best approaches.

Due to improvements of the discipline, machine learning techniques are used and applied in many different areas, for instance in medical diagnosis or in biology. Convolutional neural networks (CNNs) and other classic predictors are adopted for assisting researchers and practitioners in better identifying objects in images. This is the case, for instance, of skin segmentation or butterfly identification. However, a drawback of this technology is that a huge amount of data is needed to train these systems, but labeled data is a scarce resource in many domains. This is one of the reasons why big efforts are spent building and publishing datasets in specific areas, such an example is Kvasir-SEG [8], a recent dataset that contains polyp images annotated at pixel level by a group of experts.

A novel architecture came from the scope of natural language processing (NLP), where researchers study how to comprehend the semantic of texts with the purpose of automating tasks such as summarization or translation. This new model called Transformer is designed with a self-attention mechanism that enables the system to focus on specific part of the input. Transformers have also been applied to computer vision tasks, gaining performance comparable to or even better than CNNs. Once again, the main drawback of these models resides in the high demand of data useful to train a stable and performing system. TransFuse [9] and UACANet [10] are two recent approaches in the medical domain that combine different techniques: the first is a combination of CNN kernels and Transformers, while the second blends U-Net and a parallel axial attention autoencoder. No matter the architecture, the aim is to capture information at both local and global levels.

As previously noticed, semantic segmentation becomes of main importance in many contexts. Autonomous vehicles, for instance, use semantic segmentation to identify objects in the environment surrounding the vehicle in order to make safe decisions[11]. In clinical practice, it helps reduce the exposure to serious risks by detecting pathologies in their early stages, such as polyps detection that may prevent the evolution of colorectal cancers [2]. Similarly, in skin detection, deep learning methods are employed in various areas, spanning from face detection to hand gesture recognition[12]. In this context, deep approaches have faced some difficulties, such as the clutter in the background that hinders the reliable detection of hand gestures in real-world environments.

CNNs have shown their appeal also in this context, two examples are the works by Roy et al. [13] and Arsalan et al. [14]. In the former work, authors suggest using a CNN based on skin detection techniques to enhance the hand detector output. The latter instead introduced a residual skip connections (OR-Skip-Net) CNN that decreases the computational effort of the network and at the same time tackles demanding skin segmentation tasks. The goal is achieved

by moving data to the last layer of the network directly from the initial layer. CNNs are also employed for automatically translate sign language [15].

A comparative analysis is reported in [12] through an extensive empirical evaluation of several leading technologies on a set of skin detection benchmarks.

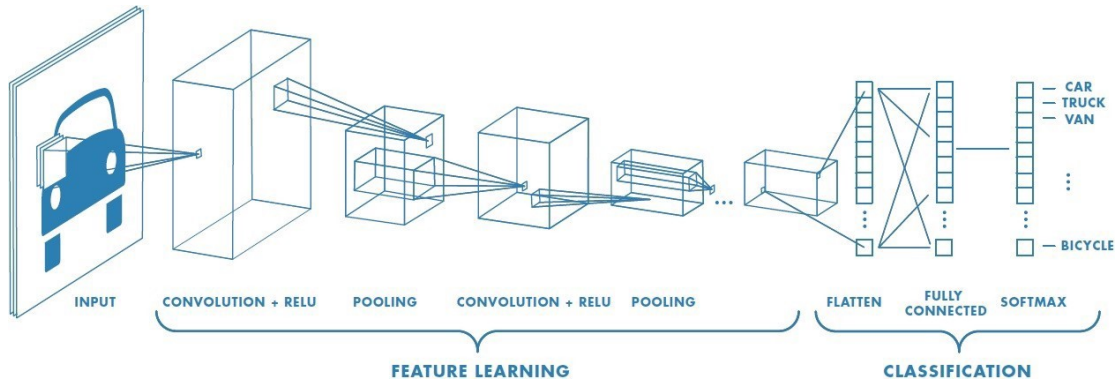


Figure 2: Example of a Convolutional Neural Network (CNN)

Recently, deep learning has also been used for the automatic recognition and classification of leukocytes [16]. This practice helps medical practitioners diagnose various blood-related diseases. This can be done in many different ways: practitioners can analyse the percentages through the histogram-based technique [15] or iterative algorithms (such as GrabCut [17]) that segment white blood cells.

Contribution: This paper proposes a new ensemble method based on DeepLabV3+, HarDNet-MSEG, and Pyramid Vision Transformers backbones. The proposal is intended to deal with semantic segmentation. In this work, diversity among individual classifiers in the ensemble is enforced by adopting different loss functions and testing different data augmentation approaches. We tested the proposed method on five different scenarios and compared the results with the existing frameworks. The empirical evaluation highlights our results that are close to or even better than the state-of-the-art level.

Chapter 2

Materials and Methods

In this section, we will provide all the techniques and approaches used to generate our ensemble. In particular, we will report the mathematical formalisation of the loss functions adopted to design the networks.

2.1 Deep Learning for Semantic Image Segmentation

In literature, different deep learning models are proposed to address semantic segmentation problems.

Semantic segmentation intends to identify objects in an image, with their corresponding boundaries. The main purpose is therefore to assign a class at the pixel level; a task achieved thanks to FCNs (Fully convolutional networks). FCNs have very high performance and unlike CNNs they use a fully convolutional last layer instead of a fully connected one [18].

In order to obtain deconvolutional networks, such as U-Net, FCNs and autoencoders are combined together.

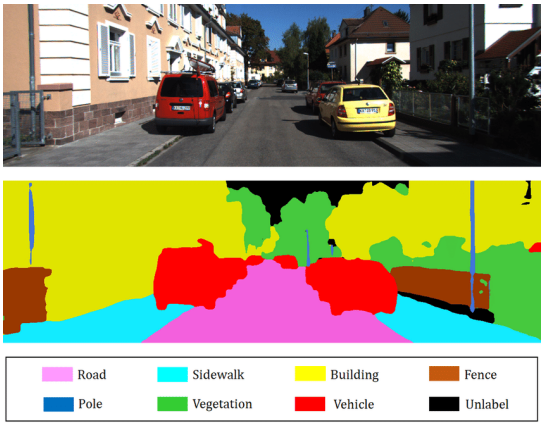


Figure 3. Semantic Segmentation of a 2D image

U-Net represents the first attempt to use autoencoders in an image segmentation task. Through the autoencoder it is possible to downsample the input and simultaneously increment the number of features used to describe the input space. We can find another symbolic example in SegNet [19]: here, the max pool indices of the relative encoder level feed the decoders, while VGG is used for encoding. This allows to reduce memory usage and also has a more promising segmentation.

DeepLab [20] consists of a series of autoencoder models introduced by Google, which has shown excellent results in semantic segmentation applications [21]. These are some of the main features introduced to guarantee good performance:

- A dilated convolution reduces the effects of pooling and stride, thereby greatly increasing the resolution.
- Through an Atrous Spatial Pyramid Pooling, information is obtained at various scales.
- A union of CNNs and probabilistic graphic models makes it possible to detect the boundaries of objects.

We find in DeepLabV3 two most important innovations: a 1x1 convolution in Atrous Spatial Pyramid Pooling and a batch normalisation: a set of modules placed in parallel and in cascade for convolutional dilation. DeepLabV3+ [5], an expansion of the family developed by Google, is adopted in this work. This expansion includes, among the most important features, a decoder with depth-wise convolutions and point-wise convolutions. The depth-wise works in the same location but with various channels, while the point-wise uses the same channel in various locations. In order to obtain different designs for a framework, we can consider other characteristics of the structure of a model.

In this paper, we will investigate ResNet101 [22], a very famous CNN that acquires a residual function by referring to the block input ([33] is recommended for an exhaustive list of CNN structures). We adopt the ResNet101 network, pre-trained on the VOC segmentation dataset and then modulated through the parameters suggested[1]. These parameters have not been modified in order to prevent overfitting phenomena (i.e. same parameters in all the tested datasets):

- initial learning rate = 0.01;
- number of epoch = 10 (using the simple data augmentation approach) or 15 (using more complex data augmentation approach due to the slower convergence using this larger augmented training set);
- momentum = 0.9;

[1] https://github.com/matlab-deep-learning/pretrained-deeplabv3plus

- L2Regularization = 0.005;
- Learning Rate Drop Period = 5;
- Learning Rate Drop Factor = 0.2;
- Shuffle training images every-epoch;
- Optimizer = SGD (stochastic gradient descent).

Firstly, we propose an ensemble of DeepLabV3+ models obtained by applying various loss and data augmentation methods, and then we combine the ensemble with HarDNet-MSEG [6], and Pyramid Vision Transformers (PVT) [7]. The HarD-Net-MSEG (Harmonic Densely Connected Network), a model influenced by Densely Connected Networks, allows the reduction of memory consumption in this way: it decreases most of the connection layers at the DenseNet level, in order to reduce the costs of concatenation. In addition, the input / output channel ratio is equalised thanks to the increase in the channel width of the layers (consequently to the increase in its connections).

The PVT is a pure convolution-free transformer network which aims to acquire a high-resolution representation starting from a fine-grained input. The computational cost of the model is decreased by a progressive pyramidal shrinkage, accompanied by the depth of the model. A spatial-reduction attention (SRA) layer is introduced to an additional reduction of the computational complexity of the system.

In this work, both HarD-Net-MSEG and PVT have been trained using the same options in all the problems: *batch size 15; number of epochs 100; initial learning rate 0.0001; decay rate=0.1; decay epoch=50*.

2.2 Loss functions

The main goal of a neural network would be to map every item perfectly by using the perfect weights. However, this is not possible due to the presence of too many unknowns.

The problem of learning is cast as an optimization where weights are modified little by little in order to make more accurate predictions. In a CNN, weights are learned by using the stochastic gradient descent algorithm.

The gradient descent algorithm seeks the direction opposite to the gradient, hence the direction along which the loss function decreases the most.

We can see neural networks as an optimization algorithm which wants to minimise the error (or the loss) of our mapping. The objective function is called loss function.

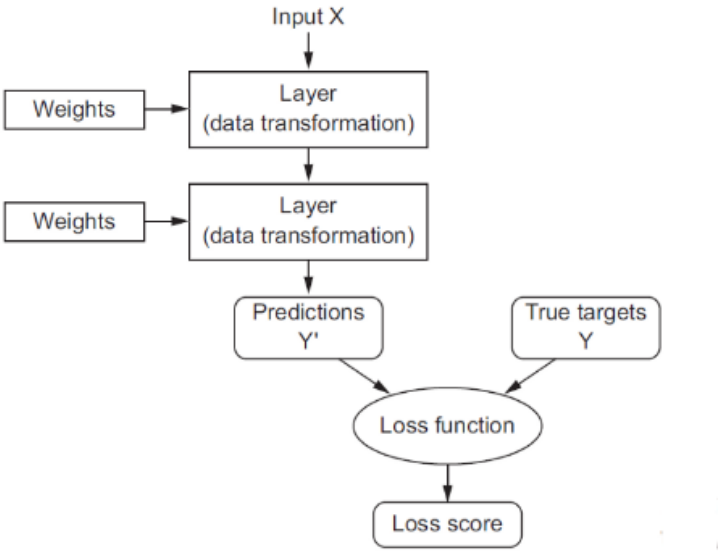


Figure 4. A loss function measures the quality of the output

Some loss functions tested for designing the networks ensemble will be presented in this section, including the loss functions suggested in [24] (subsection 2.2.1-2.2.7) (tested also in [24]) and the new ones here proposed.

Different loss functions influence the training phase and the performance of the model. In semantic segmentation tasks, pixel-wise-cross-entropy is one of the most widespread and adopted loss functions; this operates at the pixel level, verifying whether the predicted label of a given pixel coincides with the ground-truth. One of the main problems with this approach is the critical situation in which the dataset is unbalanced with respect to the labels, but it can be solved through the use of counterweights. The Dice loss function [25] aims to verify the overlap between the predicted segmented images and the ground-truth. This approach, which has also been used in this work, is widespread in semantic segmentation.

An exhaustive overview of image segmentation and loss functions is available in a recent survey [25].

2.2.1 Dice Loss

A widely adopted metric for evaluating the performance of models used for semantic segmentation is the Dice Loss, obtained from the Sørensen-Dice coefficient. This coefficient made it possible to evaluate how similar two images are and their value span in the interval [0, 1]. In [26] Generalized Dice Loss was introduced, a multiclass variant of Dice Loss.

We denote the Generalized Dice Loss between the predictions Y and the training targets T as:

$$L_{GD}(Y,T) = 1 - \frac{2 * \sum_{k=1}^{K} w_k * \sum_{m=1}^{M} Y_{km} * T_{km}}{\sum_{k=1}^{K} w_k * \sum_{m=1}^{M} (Y_{km}^2 + T_{km}^2)} \quad (1)$$

$$w_k = \frac{1}{\left(\sum_{m=1}^{M} T_{km}\right)^2} \quad (2)$$

Here, M represents the number of pixels, K represents the number of classes.

The aim of the weighting factors w_k is to facilitate the network to concentrate on a limited region (therefore it is inversely proportional to the labels frequency of a given class k).

2.2.2 Tversky Loss

A frequent issue in machine learning as well as in image segmentation is represented by unbalanced classes, i.e., the phenomenon whereby one class prevails over another. To solve this problem, Tversky Loss function was proposed [27]. The original formula of the Tversky index, an expansion of the Dice similarity coefficient that can help us formalise the loss function, is the following:

$$TI_k(Y,T) = \frac{\sum_{m=1}^{M} Y_{pm} T_{pm}}{\sum_{m=1}^{M} Y_{pm} T_{pm} + \alpha \sum_{m=1}^{M} Y_{pm} T_{nm} + \beta \sum_{m=1}^{M} Y_{nm} T_{pm}} \quad (3)$$

Here, T represents the ground truth for a certain class k, Y represents the predictions; α and β are two weighting factors used to handle a trade-off between false negatives and false positives; M indicates the total number of pixels, n represents the negative class and p the positive class. A particular case is when $\alpha = \beta = 0.5$, we have that the Tversky Index boils down to the Dice Similarity coefficient.

We can formalise, based on the aforementioned formula, the Tversky Loss as:

$$L_T(Y,T) = \sum_{k=1}^{K} (1 - TI_k(Y,T)) \quad (4)$$

Here, K is the number of classes.

In our code, we fix $\alpha = 0.3$ and $\beta = 0.7$. We use these values in order to focus on false negatives.

2.2.3 Focal Tversky Loss

The CE function (cross-entropy) intends to limit the dissimilarity between two probability distributions. Several versions of CE can be found in the literature, including for example focal loss [28] and binary cross entropy. The first, using a modulating factor $\gamma > 0$, consents the model to focus on hard samples instead of properly classified examples. The second is an adaptation of CE that must be applied to binary classification problems (i.e., only-two classes problems).

Focal Tversky Loss is formalised as:

$$L_{FT}(Y,T) = L_T(Y,T)^{\frac{1}{\gamma}} \tag{5}$$

In our work, we choose $\gamma = 4/3$.

2.2.4 Focal Generalized Dice Loss

Moreover, the modulating factor was used in Generalized Dice Loss obtaining the
Focal Generalized Dice Loss [29], a function that by focusing on very limited Regions of Interest allows to decrease the weight of common samples.

$$L_{FGD}(Y,T) = L_{GD}(Y,T)^{\frac{1}{\gamma}} \tag{6}$$

In our work, we choose $\gamma = 4/3$.

2.2.5 Log-Cosh Type Losses

By combining Dice Loss and Log-Cosh function we obtain Log-Cosh Dice Loss. Log-Cosh function is commonly applied with the purpose of smoothing the curve in regression applications. Actually, for small x, $\log(\cosh(x))$ corresponds to $x^2/2$ and for large x to $|x| - \log(2)$. Log-Cosh Generalized Dice Loss is formalised as:

$$L_{lcGD}(Y,T) = \log(\cosh(L_{GD}(Y,T))) \tag{7}$$

With the intention of smoothing their curves the same logic has been applied to other loss functions, e.g. the Log-Cosh Focal Tversky Loss, which we can formalised as:

$$L_{lcFT}(Y,T) = \log(\cosh(L_{FT}(Y,T))) \tag{8}$$

2.2.6 SSIM Loss

SSIM Loss [30] is obtained from the Structural similarity (SSIM) index [31], usually adopted to evaluate the quality of an image. SSIM index can be formalised as:

$$SSim(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (9)$$

Here, μ_x, μ_y are the local means, σ_x, σ_y, are the standard deviations, and σ_{xy}, is the cross-covariance for images x, y, while C_1, C_2 are regularization constants.

The SSIM Loss between one prediction Y and the corresponding training target T is defined as:

$$L_S(Y,T) = 1 - SSim(Y,T) \quad (10)$$

Moreover, here we proposed $L_{MS}(Y,T)$, defined as L_S but instead of SSIM we use the Multiscale structural similarity (MS-SSIM) index.

2.2.7 Different Functions Combined Loss

The possibility of obtaining high precision but low recall is frequent with unbalanced data.

In Generalized Dice Loss a common strategy is applied to mitigate the effects of the class imbalance. We define as weight w_k the inverse of the frequency of the label.

To reduce the network probability of missing lesions, in certain contexts it is advisable to weight false positives lower than false negatives. With the intention of emphasising the benefits of both loss functions and increasing the capacity of the model to focus on difficult samples, we combine Focal Tversky Loss and Generalized Dice Loss. The formula of the combination is:

$$Comb_1\ (Y,T) = L_{FGD}(Y,T) + L_{FT}(Y,T) \quad (11)$$

In the same way, combining Log-Cosh Dice Loss, Focal Generalized Dice Loss, and Log-Cosh Focal Tversky Loss allows to reduce the weight of simple samples. Furthermore, we control the non-convex behavior of the curve by adopting the Log-Cosh function:

$$Comb_2(Y,T) = L_{lcGD}(Y,T) + L_{FGD}(Y,T) + L_{lcFT}(Y,T) \quad (12)$$

In our experiments we also tried to combine Generalized Dice Loss and the SSIM Loss:

$$Comb_3\ (Y,T) = L_S(Y,T) + L_{GD}(Y,T) \quad (13)$$

2.2.8 Cross Entropy

The cross-entropy (CE) loss function provides us with a measure of the difference between two probability distributions. The goal is to minimize this difference and, in doing so, it has no bias between small or large regions.

This could be an issue when dealing with imbalanced datasets. Hence, the weighted cross-entropy loss was introduced and it resulted in well-balanced classifiers for imbalanced scenarios [32].

The formula for weighted binary cross entropy is presented in (14). In this equation, T refers to the ground truth label image, while T_{ik} is the true value for the pixel i and it can be equal to either 0 or 1. It is equal to 1 if the pixel i belongs to the class k, 0 otherwise.

P is the prediction for the output image and P_{ik} is the probability of the i-th pixel to belong to the k-th class obtained by using the sigmoid activation function. For P we used the softmax activation function, which returns probabilities.

w_{ik} is the weight given to the i-th pixel of the image for the class k. These weights were calculated by using an average pooling over the mask with a kernel 31x31 and a stride of 1 in order to consider also nonmaximal activations.

$$L_{WBCE} = -\sum_{k=1}^{K}\sum_{i=1}^{N} w_{ik} * T_{ik} * \log(P_{ik}) \tag{14}$$

Where K is the number of classes and N the number of pixels.

2.2.9 Weighted Intersection over Union

Another well-known loss function is Intersection over Union (IoU) loss, which was introduced for the first time in [33]. The original formula was:

$$IoU = \frac{|P \cap T|}{|P \cup T|} \tag{15}$$

As mentioned earlier, T is the truth label image and P is the prediction for the output image. Unfortunately, the set symbols for Intersection and Union are not differentiable because P and T have to be either 0s or 1s. This is not true for P, so the formula was then approximated with the following:

$$IoU' = \frac{|P \times T|}{|P + T - P \times T|} \tag{16}$$

Where $P \times T$ is the element-wise multiplication of T and P. For what concerns the denominator, we subtract the "intersection" between P and T because we do not want to consider the intersection twice.

Once the set operators have been converted into arithmetic ones, the formula is differentiable and it is possible to evaluate the gradient.

However, IoU is an evaluation metrics used for evaluating the goodness of the prediction. Hence, a value of 1 is equivalent to a perfect prediction. For this reason, the loss function will be:

$$L_{IoU} = 1 - IoU' \tag{17}$$

Unfortunately, this function has to face the same problem of CE in inferring the label of the boundary of each object, therefore, as suggested in [34], we use the weighted Intersect over Union (wIoU), instead of the standard IoU.

The formula of the weighted Intersect over Union loss will be:

$$L_{wIoU} = 1 - \frac{|w * P * T|}{|w * (P + T) - w * P * T|} = 1 - \frac{\sum_{i=1}^{N} w_{ik} * \sum_{k=1}^{K} T_{ik} * Y_{ik} + 1}{\sum_{i=1}^{N} \sum_{k=1}^{K} w_{ik}(T_{ik} + Y_{ik} - T_{ik} * Y_{ik}) + 1} \tag{18}$$

Where N is the number of pixel and K is the number of classes. The weights w_{ik} are calculated as said previously. T_{ik} and Y_{ik} are, respectively, the ground truth value and the prediction value for the pixel i belonging to the class k. We added 1 to both the numerator and the denominator in order to prevent the undefined division $\frac{0}{0}$.

2.2.10 Structure Loss

Now, based on the intuition in [6], weighted Intersect over Union and weighted binary-crossed entropy are considered together.

$$L'_{STR} = L_{wIoU} + L_{wbce} \tag{19}$$

We decided to change the proposed loss in the following way:

Instead of applying an avgpool over the mask, we have done this over the predictions to improve the diversity in the deep neural network.

Then, we want to give more importance to the binary-crossed entropy loss, so we use a weight of 2 for that one.

We obtain the final loss, which is:

$$L_{STR} = L_{wIoU} + 2L_{wbce} \tag{20}$$

2.2.11 BoundExpStructure

We decided to combine Structure Loss, Boundary Loss and Exponential Logarithmic Loss in order to have better performances on the small structures of a highly imbalanced dataset and, at the same time, have better identification of the boundaries of the image.

$$L_{BoundExpS} = L_{Bound} + L_{Exp} + L_{Str} \tag{21}$$

2.2.12 Boundary Enhancement Loss

The Boundary Enhancement Loss is a loss proposed in [35] which explicitly focus on the boundary areas during training.

This loss has very good performance as it requires neither any pre- nor post-processing of the image nor a particular net in order to work.

The Laplacian filter $\mathcal{L}(\cdot)$ is the milestone of this loss as it generates strong responses around the boundaries and zero everywhere else. When applying the laplacian filter to a mask S, the result is:

$$\mathcal{L}(\mathrm{x},\mathrm{y}) = \frac{\partial^2 S}{\partial x^2} + \frac{\partial^2 S}{\partial y^2} \tag{22}$$

The positive aspect about using the Laplacian filter, is that it can be achieved quite easily through a series of convolution operations. As a result, the idea is to evaluate the difference between the filtered output of ground truth labels and the filtered output of the predictions.

The boundary enhancement loss, as proposed in [35] is:

$$L_{BE} = \left|\left|\mathcal{L}(T) - \mathcal{L}(Y)\right|\right|_2 = \left|\left|\frac{\partial^2 (T-Y)}{\partial x^2} + \frac{\partial^2 (T-Y)}{\partial y^2}\right|\right|_2 \tag{23}$$

Where $||\cdot||_2$ is the l_2 norm. This can be easily achieved as already described in the original paper [35].

Based on the idea of the same paper, we used Dice Loss and Boundary Enhancement loss together, weighted appropriately, and the Structure Loss:

$$L_{DiceBES} = \lambda_1 L_{Dice} + \lambda_2 L_{BE} + L_{Str} \tag{24}$$

The best results were achieved by using $\lambda_1 = 1$ and $\lambda_2 = 0.01$

2.2.13 Contour-aware Loss

Contour-aware Loss was proposed for the first time in [74]. It consists in a weighted binary cross-entropy loss where the weights are obtained with the aim of giving more importance to the borders of the image.

In the loss a morphological gradient edge detector was employed. Basically, the difference between the dilated and the eroded label map is evaluated. For smoothing purposes, the Gaussian blur was later applied. This spatial weight map can be formulated as:

$$M^C = Gauss\left(K \cdot \left(dilate(T) - erode(T)\right)\right) + \mathbb{1} \tag{25}$$

here $dilate(T)$ and $erode(T)$ are dilation and erosion operations with a 5 × 5 kernel. K is a hyperparameter for assigning the high value to contour pixels which was set to 5 empirically. $\mathbb{1}$ is the matrix with 1 in every position.

We can formalise now the new loss:

$$L_C = -\sum_{i=1}^{N} M_i^C * (T_i * \log(Y_i) + (1 - T_i) * \log(1 - Y_i)) \tag{26}$$

Finally, the we are going to use in our ensembles is:

$$L_{CS} = L_C + L_{Str} \tag{27}$$

2.3 Data Augmentation

The training phase of a classifier and the resulting performance of the system are strongly influenced by the size of the dataset. This is also true for an ensemble method. Thus, to increase the amount of data that can be used to train the system, several techniques may be applied to the original dataset. In the next paragraphs, we shall describe the different techniques adopted with the purpose of data augmentation. We employ these techniques on the training set, both on the input samples and their mask. We leave the test set unchanged.

Two different data augmentation approaches are tested:

- DA1, base data augmentation consisting in horizontal and vertical flip, 90° rotation.
- DA2, the following operations are performed:
    1. The image is displaced to the right or the left.
    2. The image is displaced up or down.
    3. The image is rotated by an angle randomly selected from the range [0°, 180°].
    4. Horizontal or vertical shear by using the Matlab function *randomAffine2d()*.

5. Horizontal or vertical flip.
6. Change in the brightness levels by adding the same value (random value between 25 and 50) to each RGB channel.
7. Change in the brightness levels by adding different values (random value between 25 and 50) to each RGB channel.
8. Add speckle noise, it adds multiplicative noise to the image I adding a value n×I, where n is uniformly distributed random noise with mean 0 and variance 0.05.
9. Application of the technique "Contrast and Motion Blur", described below.
10. Application of the technique "Shadows", described below.
11. Application of the technique "Color Mapping", described below.

Some artificial images (DA2 approach) contain only background pixels; to discard them we simply delete all the images where there are less than 10 pixels that belong to the foreground class.

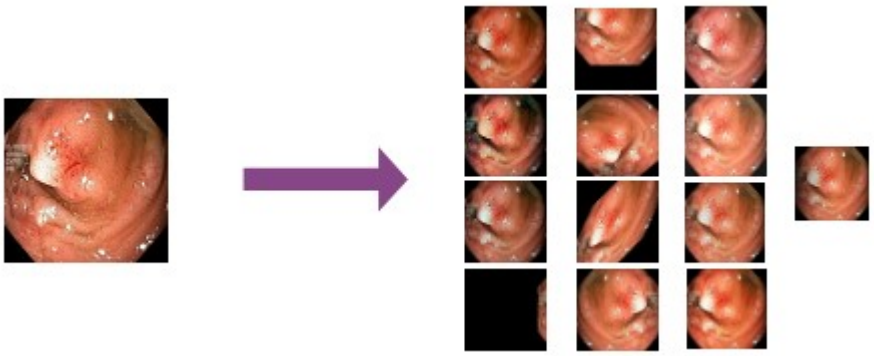


Figure 5: Data Augmentation DA2

2.3.1 Shadows

New image samples can be obtained by creating shadows in the original set of images. Shadows may be created randomly to the left or to the right of the original image. We use the following criteria to decide the intensity of the shadow (*direction* = 1: right; *direction* = 0: left):

$$y = \begin{cases} min\left\{0.2 + 0.8\sqrt{\frac{x}{0.5}},\ 1\right\} & direction = 1 \\ min\left\{0.2 + 0.8\sqrt{\frac{1-x}{0.5}},\ 1\right\} & direction = 0 \end{cases} \tag{28}$$

2.3.2 Contrast and Motion Blur

Another technique for data augmentation that allows to derive new samples from an original dataset is based on the combination of contrast and motion blur. The first one increases or decreases the original contrast, the second one simulates the movement of the camera taking the image. We developed two different contrast function and each time we choose one of them at random.

The first function is defined as follows:

$$y = \frac{\left(x - \frac{1}{2}\right)\sqrt{1 - \frac{k}{4}}}{\sqrt{1 - k(x - \frac{1}{2})^2}} + 0.5, \qquad k \leq 4 \tag{29}$$

The parameter k controls the contrast. Specifically: The contrast is increased when $k < 0$; it is decreased when $0 < k \leq 4$; the image is unchanged when $k = 0$.

The value of the parameter is drawn at random in the following range:

$\mathcal{U}(2.8, 3.8)$ → Hard decrease in contrast;

$\mathcal{U}(1.5, 2.5)$ → Soft decrease in contrast;

$\mathcal{U}(-2, -1)$ → Soft increase in contrast;

$\mathcal{U}(-5, -3)$ → Hard increase in contrast.

The second function is defined as follows:

$$y = \begin{cases} \frac{1}{2}\left(\frac{x}{0.5}\right)\alpha & 0 \leq x < \frac{1}{2} \\ 1 - \frac{1}{2}\left(\frac{1-x}{0.5}\right)\alpha & \frac{1}{2} \leq x \leq 1 \end{cases} \tag{30}$$

The parameter α controls the contrast. In particular, the contrast is increased when $\alpha > 1$; it is decreased when $0 < \alpha < 1$; if $\alpha = 1$, then the image is left unchanged.

The parameter is chosen randomly from four possible ranges:

$\mathcal{U}(0.25, 0.5)$ → Hard decrease in contrast;

$\mathcal{U}(0.6, 0.9)$ → Soft decrease in contrast;

$\mathcal{U}(1.2, 1.7)$ → Soft increase in contrast;

$\mathcal{U}(1.8, 2.3)$ → Hard increase in contrast;

The blurring effect that mimics the movement of the camera is applied right after the contrast adjustment. We use the MATLAB function *fspecial('motion', len, theta)*.

2.3.3 Color Mapping

Changing the color map of the image produces a new image. In particular, it is possible to map the color of an image to the one of another image. We generated a pair of images by coupling any image in the original training set with another randomly selected image in the same set. We adopted the Stain Normalization toolbox[2] which provides this functionality in three different versions:

- RGB Histogram Specification
- Reinhard
- Macenko

[2] The toolbox is authored by Nicholas Trahearn and Adnan Khan and available online at https://warwick.ac.uk/fac/cross_fac/tia/software/sntoolbox/

Chapter 3

Results

We run an extensive empirical evaluation with the aim of measuring the performance of our ensemble. We comprehend a comparison with several state-of-the-art models for a more exhaustive evaluation of our system. The empirical evaluation is carried out on five real-world scenarios: polyp segmentation, skin segmentation, leukocyte identification, butterfly and microorganism identification.

3.1 Metrics

The system has been evaluated using two standard metrics: Dice score and Intersection over Union (IoU). In the following formulae, TP, TN, FP, FN correspond to the true positives, true negatives, false positives, and false negatives, respectively. A is the predicted mask (TP+FP) and B is the ground truth map (TP+FN).

Dice score (which is equivalent to F1score in binary classification tasks) is a weighted average of precision and recall. Formally, it is defined as:

$$F1score = Dice = \frac{|A \cap B|}{|A| + |B|} = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{31}$$

Intersection over Union (IoU) defines the shared area between two masks, divided by the area of the union between the two maps. Formally, it is defined as:

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{32}$$

In the experiments, images size has been modified due to the input size of the models. In these cases, the predicted masks have always been changed back to their original dimension.

3.2 Datasets and testing protocols

Some examples for images and masks, from each of the five datasets, are displayed in Figure 6. It is clear that they are very different segmentation problems.

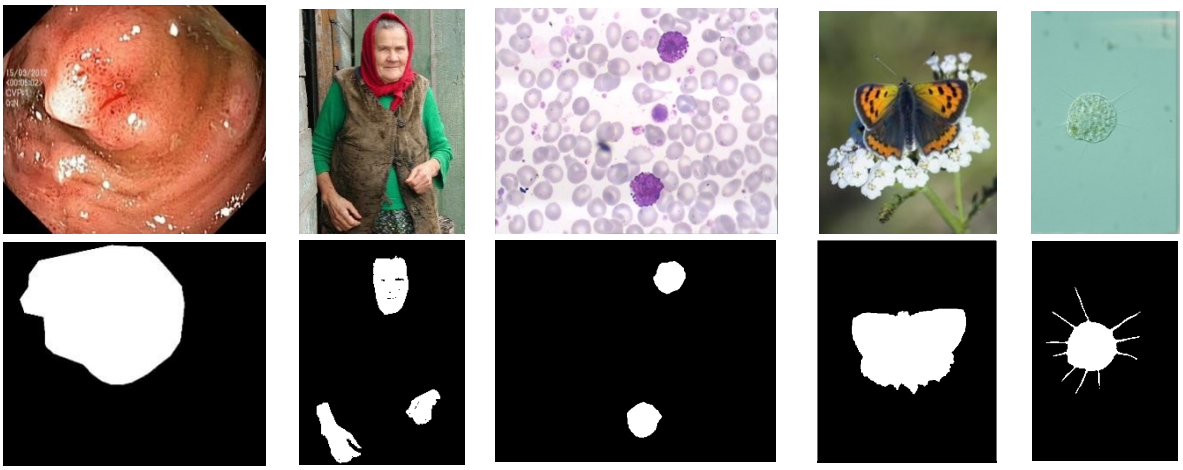


Figure 6. Samples from polyp segmentation, skin segmentation, leukocyte identification, butterfly and microorganism identification (images and masks).

3.2.1 Polyp segmentation (POLYP)

Polyp segmentation from colonoscopy is a challenging task requiring a two-class classification between the low contrast colon background and polyp foreground pixels.

We present experimental results according to a very popular benchmark [6] available on GitHub[3] and including five datasets for polyp segmentation (i.e. Kvasir [37], ColonDB [38], CVC-T [39] and ETIS [40], ClinicalDB [41])): the training set is made by 1450 images (from the largest datasets, i.e. 900 images from Kvasir and 550 images from ClinicalDB) the others are for the test set (100 images from Kvasir, 380 from ColonDB, 60 from CVC-T, 196 from ETIS and 62 from ClinicalDB), as usually done in the literature. In these datasets we use resized images of size 352×352.

[3] https://github.com/james128333/HarDNet-MSEG

3.2.2 Skin segmentation (SKIN)

The segmentation task in skin detection consists in recognizing the image portions that represent "skin" and "no skin": as a result, it is a binary classification problem. We use the framework proposed in [12] in this paper, which includes a small training set of 2000 images from the ECU dataset [42] and ten datasets very different to each other, see Table 1. According to the original testing protocol [12], dice (i.e. F1-score) is calculated at the pixel level and not at the image level and averaged for the whole dataset. In these datasets we use resized images of size 352×352.

Table 1. Test skin datasets. ECU dataset is split: 2000 images for training and 2000 for test.

ShortName	Name	#Samples	Ref.
Prat	Pratheepan	78	[43]
MCG	MCG-skin	1000	[44]
UC	UChile DB-skin	103	[45]
CMQ	Compaq	4675	[46]
SFA	SFA	1118	[47]
HGR	Hand Gesture Recognition	1558	[48]
Sch	Schmugge dataset	845	[49]
VMD	Human activity recognition	285	[50]
ECU	ECU Face and Skin Detection	2000	[42]
VT	VT-AAST	66	[51]

3.2.3 Leukocyte segmentation (LEUKO)

Leukocyte recognition is the task of segmenting the white blood cells from the background, with the aim of diagnosing many diseases such as leukemia and infections. In our experiment we used the freely available[4] LISC database [16], a collection of 250 hematological images extracted from the peripheral blood of eight healthy people. Images have been acquired at high resolution (720×576 pixels) and manually labelled to segment 10 different types of leukocytes. In this work, we do not perform classification, therefore we consider only segmentation performance. The testing protocol, as suggested by the authors of the dataset, is a 10-fold cross

[4] http://users.cecs.anu.edu.au/~hrezatofighi/Data/Leukocyte%20Data.htm

validation: Dice results are calculated at image level and averaged for each fold and then on the 10 folds. In this dataset we use resized images of size 513×513.

3.2.4 Butterfly identification (BFLY)

As already done in the literature, for butterfly identification we adopted the public dataset[5] [52]. For a fair comparison we used the same testing protocol proposed by the authors of the dataset: 4-fold cross validation, each fold includes 208 test images and 624 training images. In this dataset we use resized images of size 513×513.

3.2.5 Microorganism identification (EMICRO)

EMicro [53] is a public dataset[6] of Environmental Microorganism Image Dataset Sixth Version (EMDS-6). It is composed of 840 images: following the original paper we split the dataset and the 37.5% of the images belongs to the test set.
In this dataset we use resized images of size 513×513.

3.3 Experiments

3.3.1 Baseline ensembles

As the aim of this paper is to study approaches to increase diversity of ensembles, we report in Table 2 the performance of some baseline classifiers and ensembles based on different network architectures (all combined with the data augmentation DA1, see section 2.3). The tests reported in section 3.3.1 are all based only on the Dice loss; moreover, for sake of space, for the polyp and skin datasets we report only the average performance value among the set of datasets. Each ensemble is made up of N models (N=1 denotes a stand-alone model) which differ only for the randomization in the training process:

- RN18 a stand-alone DeepLabV3+ segmentator with backbone Resnet18 (pretrained in ImageNet);
- ERN18(N) is an ensemble of N RN18 networks (pretrained in ImageNet);
- RN50 a stand-alone DeepLabV3+ segmentators with backbone Resnet50 (pretrained in ImageNet);
- ERN50(N) is an ensemble of N RN50 networks;

[5] http://www.josiahwang.com/dataset/leedsbutterfly/
[6] https://figshare.com/articles/dataset/EMDS-6/17125025/1

- RN101 a stand-alone DeepLabV3+ segmentators with backbone Resnet101 (pretrained as detailed in the above section 2.1);
- ERN101(*N*) is an ensemble of *N* RN101 networks.

Table 2. Performance (Dice) of the proposed ensembles in the five benchmark datasets, the last column AVG reports the average performance.

	Polyp	**Skin**	**Leuko**	**Bfly**	**EMicro**	***Avg***
RN18	0.806	0.865	0.897	0.960	0.908	0.887
RN50	0.802	0.871	0.895	0.968	0.909	0.889
RN101	0.808	0.871	0.915	0.976	0.918	0.898
ERN18(10)	0.821	0.866	0.913	0.963	0.913	0.895
ERN50(10)	0.807	0.872	0.897	0.969	0.918	0.893
ERN101(10)	**0.834**	**0.878**	**0.925**	**0.978**	**0.919**	**0.907**

The results in Table 2 show that, although the overall performance increases when switching from the stand-alone version to an ensemble, the improvement is not as high as one might expect, indicating that the individual approaches are quite stable. Maybe this result is related to the architecture of the DeepLabV3+ network: its internal modules apply atrous convolutions in cascade or in parallel to capture multi-scale context by adopting several atrous rates. This solution, which has been designed to solve the problem of segmenting objects at multiple scales, also mimics an ensemble approach thanks to the fusion of activations taken at different levels of the encoder, making the resulting segmentation quite stable.

The best method is to use ResNet101 as backbone.

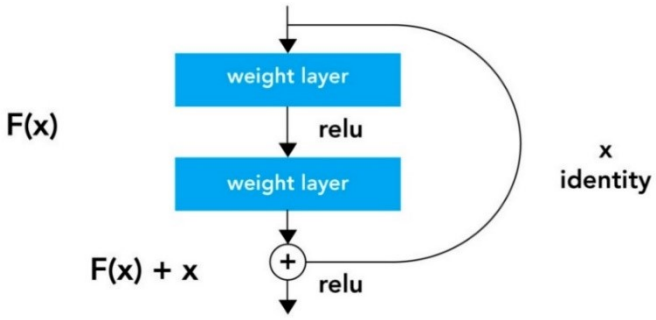


Figure 7. A building block of a ResNet (Residual Network). Several of these blocks are stack on top of one another. The innovation here is the presence of the 'skip connection' (on the right-hand side) which resulted in networks with better accuracy than networks with plain layers as the output is the result of the input both unchanged and multiplied by the weights. A ResNet101 has 101 weight layers.

3.3.2 Ablation studies

The first ablation study is related to the evaluation of different loss functions to increase the diversity of the models and improve the performance of the ensemble. In Table 3 the performance of RN101, with the different loss functions here tested/proposed, is reported and compared to the dice loss as baseline and DA1 as data augmentation method. For sake of space, for the polyp and skin datasets we report only the average performance value among the set of datasets. The stand-alone networks are later fused, always using the sum rule, in some ensembles:

- ELoss101(10) is an ensemble, combined by sum rule, of 10 RN101 each coupled with data augmentation DA1 and a given loss function, the final fusion is given by: $2\times L_{GD}+ 2\times L_T+ 2\times$ Comb1 + $2\times$ Comb2+$2\times$Comb3; where with $2\times L_{GD}$ we mean two RN101 trained using L_{GD} loss function.
- ELossMix(10) is an ensemble similar to the previous one, but here data augmentation is used to increase diversity: the networks coupled with the loss used in ELoss101(10) (L_{GD}, L_T, Comb1, Comb2, Comb3) are trained one time using DA1 and another time using DA2 (i.e. 5 networks each trained twice, hence we have an ensemble of 10 networks);
- ELossLarge(10) is an ensemble of 10 networks, the networks trained using (L_{GD}, L_T, Comb1, Comb2, Comb3) use DA2 as augmented training set, while the networks trained

considering the new loss functions tested in this work (L_{STR}, $L_{BoundExpS}$, $L_{DiceBES}$, L_{MS}, L_{CS}) are coupled with DA1.

Table 3. Performance (Dice) of some stand-alone methods and ensembles in the five benchmark datasets when varying the loss function, the last column AVG reports the average performance.

	LOSS	Polyp	Skin	Leuko	BFly	EMicro	*Avg*
RN101	L_{GD}	0.808	0.871	0.915	0.976	0.918	0.898
RN101	L_{STR}	0.809	0.869	0.930	0.964	0.901	0.895
RN101	$L_{BoundExpS}$	0.803	0.874	0.928	0.978	0.901	0.897
RN101	$L_{DiceBES}$	0.819	0.869	0.922	0.969	0.904	0.897
RN101	L_{MS}	0.813	0.860	0.920	0.972	0.920	0.897
RN101	L_{CS}	0.823	0.873	0.917	0.967	0.911	0.898
ERN101(10)	L_{GD}	0.834	0.878	0.925	0.978	0.919	0.907
ELoss101(10)	Many loss	0.842	0.880	0.925	0.980	0.921	0.910
ELossMix(10)	Many loss	**0.851**	**0.883**	0.936	0.983	**0.924**	0.915
ELossLarge(10)	Many loss	0.848	**0.883**	**0.944**	**0.984**	0.922	**0.916**

The results reported in Table 3 show that the proposed new loss functions gain performance similar to Dice and can be considered a useful starting point for the design of an ensemble. In fact, the good performance of *ELoss101* and *ELossLarge* with respect to ERN101(10) proves that the including networks trained by different loss functions are useful for the creation of an ensemble: this observation is even more evident considering that it is validated on very different problems.

We should not overlook the value of altering the training set, in this case through the use of different data augmentation: this seems to be the winning strategy when combined with different loss functions (*ELossMix*). Finally, we can notice that *ELossMix* and *ELossLarge* obtain similar performances.

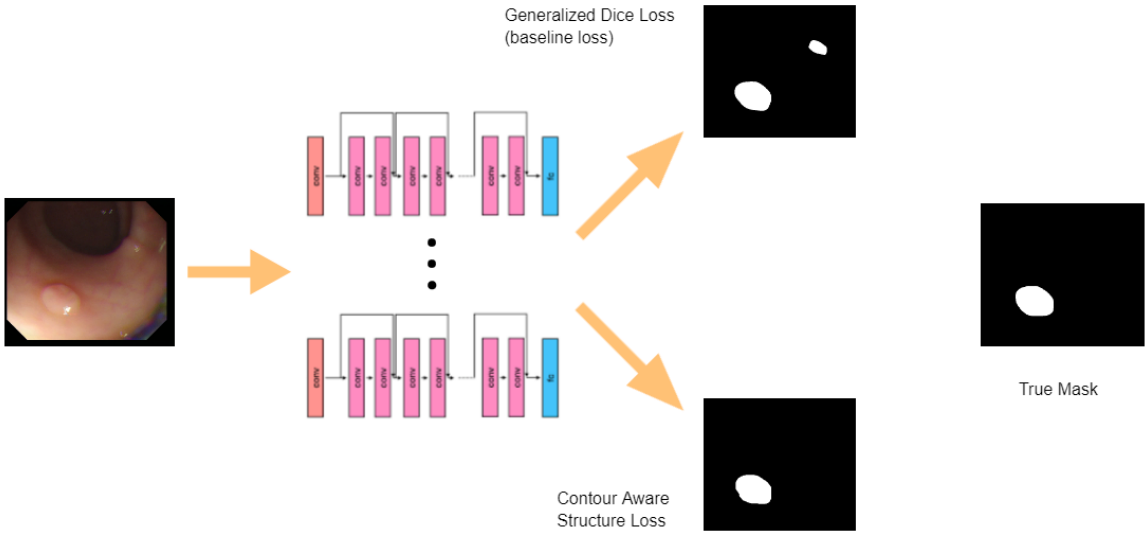


Figure 8: Comparison of the output of the network after setting the weights with 2 different losses: Generalized Dice Loss (baseline loss) and Contour Aware Structure Loss (one of the proposed loss functions).

The second ablation study is related to the evaluation of different architectures. Performance of the above cited methods coupled with different data augmentation strategies are reported in Table 4. The names DA1 and DA2 refer to the strategies explained in section 2.3, while DA1/2 denotes that the given ensemble is obtained by the fusion of networks based on DA1 and networks based on DA2.

HardNet-MSEG is trained with two different optimizers: SGD denoted as H_S and Adam denoted as H_A. The ensemble FH is the fusion of HarDNet-MSEG trained with different optimizers. PVT is trained using AdamW optimizer (as suggested in the original paper where PVT has been proposed). The loss function for both HarDNet-MSEG and PVT is the same of the original papers.

Some further ensembles are reported in table 4:

- PVT(2), sum rule between PVT combined with DA1 and PVT combined with DA2;
- FH(2), sum rule among two H_S (one combined with DA1, the latter with DA2) and two H_A (one combined with DA1, the latter with DA2);
- FH(2)+2×PVT(2), weighted sum rule between PVT(2) and FH(2), the weight of PVT(2) is assigned so that its importance in the ensemble is the same of FH(2) (notice that FH(2) consists of four networks while PVT(2) is built by only two networks).
- ELossMix(10)+(10/4)×FH(2)+(10/2)×PVT(2), weighted sum rule among ElossMix(10), FH(2) and PVT(2), as in the previous ensemble, the weights are assigned

so that each ensemble member has the same importance (notice that ElossMix(10) is the fusion by sum rule of 10 DeepLabV3+).

Table 4. Performance (Dice) of some stand-alone methods and ensembles in the five benchmark datasets.

	DA	**Polyp**	**Skin**	**Leuko**	**BFly**	**EMicro**	***Avg***
ELossMix(10)	DA1/2	0.851	0.883	0.936	0.983	0.924	0.915
H_A	DA1	0.840	0.867	0.923	0.977	0.914	0.904
H_A	DA2	0.854	0.871	0.945	0.982	0.912	0.913
H_S	DA1	0.816	0.872	0.889	0.969	0.894	0.880
H_S	DA2	0.847	0.870	0.917	0.976	0.901	0.902
FH	DA1	0.859	0.879	0.913	0.980	0.915	0.909
FH(2)	DA1/2	0.862	0.885	0.934	0.982	0.916	0.916
PVT	DA1	0.854	0.878	0.954	0.975	0.920	0.916
PVT	DA2	0.855	0.879	0.954	0.984	0.919	0.918
PVT(2)	DA1/2	0.855	0.883	0.957	0.984	0.922	0.920
FH(2)+2×PVT(2)	DA1/2	**0.875**	0.892	**0.955**	**0.985**	0.924	**0.926**
ELossMix(10)+(10/4)×FH(2)+(10/2)×PVT(2)	DA1/2	**0.875**	**0.893**	0.953	**0.985**	**0.926**	**0.926**

The results reported in Table 4 permit to draw the following conclusions:

- PVT(2), FH(2) and ElossMix(10) obtain very similar performance, only in Leuko the performance of PVT(2) is better than FH(2) and ElossMix(10);
- PVT(2) permits to obtain a very slight performance improvement with respect to stand-alone PVT, even the improvement of FH(2) with respect to the best stand-alone HardNet (i.e. H_A combined with DA2).
- Combining different architectures permits to obtain the best performance, the best trade-off "complexity vs performance" is given by "FH(2)+2×PVT(2)".

3.3.3 Comparison with the literature

For comparison with other approaches in the literature, the results of our best ensembles are reported in full in the different datasets of Polyp segmentation (Table 5) and skin detection (Table 6). The following tests clearly show that FH(2)+2×PVT(2) obtains state-of-the-art performance.

Table 5. Performance (Dice and IoU) in the polyp segmentation problem.

Method	Kvasir		ClinicalDB		ColonDB		ETIS		CVC-T		Average	
	IoU	**Dice**	**IoU**	**Dice**	**IoU**	**Dice**	**IoU**	**Dice**	**IoU**	**Dice**	**IoU**	**Dice**
FH(2)+2×PVT(2)	**0.874**	**0.920**	0.894	0.937	**0.751**	**0.826**	**0.717**	**0.787**	0.842	0.904	**0.816**	**0.875**
Ensemble in [24]	0.871	0.917	0.886	0.931	0.697	0.769	0.663	0.740	0.829	0.901	0.790	0.852
HarDNet-MSEG [6]	0.857	0.912	0.882	0.932	0.66	0.731	0.613	0.677	0.821	0.887	0.767	0.828
PraNet (from [6])	0.84	0.898	0.849	0.899	0.64	0.709	0.567	0.628	0.797	0.871	0.739	0.801
SFA (from [6])	0.611	0.723	0.607	0.700	0.347	0.469	0.217	0.297	0.329	0.467	0.422	0.531
U-Net++ (from [6])	0.743	0.821	0.729	0.794	0.41	0.483	0.344	0.401	0.624	0.707	0.570	0.641
U-Net (from [6])	0.746	0.818	0.755	0.823	0.444	0.512	0.335	0.398	0.627	0.710	0.581	0.652
SETR [21]	0.854	0.911	0.885	0.934	0.69	0.773	0.646	0.726	0.814	0.889	0.778	0.847
TransUnet [54]	0.857	0.913	0.887	0.935	0.699	0.781	0.66	0.731	0.824	0.893	0.785	0.851
TransFuse [9]	0.870	**0.920**	**0.897**	**0.942**	0.706	0.781	0.663	0.737	0.826	0.894	0.792	0.855
UACANet [10]	0.859	0.912	0.88	0.926	0.678	0.751	0.678	0.751	**0.849**	**0.910**	0.789	0.850
SANet [55]	0.847	0.904	0.859	0.916	0.670	0.753	0.654	0.750	0.815	0.888	0.769	0.842
MSNet [56]	0.862	0.907	0.879	0.921	0.678	0.755	0.664	0.719	0.807	0.869	0.778	0.834
PVT [7]	0.864	0.917	0.889	0.937	0.727	0.808	0.706	**0.787**	0.833	0.900	0.804	0.869
SwinE-Net [57]	0.870	**0.920**	0.892	0.938	0.725	0.804	0.687	0.758	0.842	0.906	0.803	0.865
AMNet [58]	0.865	0.912	0.888	0.936	0.690	0.762	0.679	0.756	---	---	---	---

Table 6. Performance (Dice=F1-score) in the skin detection problem.

	DA	Prat	MCG	UC	CMQ	SFA	HGR	Sch	VMD	ECU	VT	*Avg*
ERN101(1)	DA1	0.922	0.887	0.923	0.823	0.948	0.969	0.750	0.748	0.948	0.796	0.871
ERN101(10)	DA1	0.924	0.887	0.920	0.845	0.952	**0.971**	0.778	0.754	0.950	0.794	0.878
ELoss101(10)	DA1	0.926	0.892	0.923	0.844	**0.956**	**0.971**	0.777	0.751	0.953	0.807	0.880
ELossMix(10)	DA1/DA2	0.924	0.893	0.929	0.850	**0.956**	0.970	0.789	0.739	0.952	0.829	0.883
H_S	DA1	0.903	0.880	0.903	0.838	0.947	0.964	0.793	0.744	0.941	0.810	0.872
H_A	DA1	0.913	0.880	0.900	0.809	0.951	0.967	0.792	0.717	0.945	0.799	0.867
PVT	DA1	0.920	0.888	0.925	0.851	0.951	0.966	0.792	0.709	0.951	0.828	0.878
FH(2)+2×PVT(2)	DA1/DA2	**0.927**	**0.894**	**0.932**	**0.868**	0.954	**0.971**	0.797	**0.767**	**0.955**	**0.853**	**0.893**
[79]	DA1	0.926	0.888	0.916	0.842	0.955	**0.971**	**0.799**	0.764	0.952	0.820	0.883

In LEUKO the authors of the dataset report an IoU of 0.842, FH(2)+2×PVT(2) obtains an higher IoU of 0.916.
In EMicro the authors of the dataset report a Dice score of 0.884, FH(2)+2×PVT(2) obtains an higher Dice of 0.924.

In the BFLY many approaches have been tested (see [59]) the two best methods reported in the literature are:

a) [59] that reports an IoU of 0.950;
b) [60] that reports an IoU of 0.945.

Our suggested ensemble (i.e. FH(2)+2×PVT(2)) strongly outperforms the previous state of the art obtaining an IoU of 0.970.
Clearly the ensemble boosts the performance of the best stand-alone network (PVT combined with DA2)
The main drawback of this approach is that the best ensemble is composed by 6 networks, this means 6x RAM requirements and 6x inference time. Anyway, the inference time is very low also using an ensemble, with the current GPU architectures it is not an issue in many problems (obviously it could become one in some applications such as autonomous drive, but not in the segmentation problems faced in this paper).
E.g. a single HarDNet-MSEG runs at 86.7 / second on a GeForce RTX 2080 Ti GPU.

We have performed a further experiment to select the optimal set of models to be included in the final ensemble. We have extracted a validation set to select the best set of networks: we have considered only the two problems including many test sets: i.e. polyp and skin segmentation. In the polyp problem, the Kvasir test set has been chosen as validation set; in the skin application problem the ECU test set has been used as validation set. We have used sequential forward floating selection (SFFS) [61] for retaining the subset of networks that maximize Dice performance indicator in the validation set.
The performance of both ensembles was lower than we expected and in both datasets our best approach (i.e. FH(2)+2×PVT(2)) gained higher performance. In both cases we have faced an overfitting problem: the images in test sets are very different among each other, therefore a larger validation set, and more comprehensive of the different variations that can occur to an image, is needed for a reliable network selection.

Finally, we performed some tests using Q-statistic for further validation of our idea to build ensembles. Yule's Q-statistic [62] was conducted to demonstrate the relationship of diversity

among the networks that belong to the ensemble. After calculation, the range of Q-statistic varies from -1 to 1. For statistically independent classifiers, Q-statistic is equal to zero.

In Table 7, we report the average Q-statistic among the network of the proposed ensemble: clearly ELossMix allows for a set of networks with greater diversity than Eloss101 and ERN101. Moreover, also FH(2)+2×PVT(2) is built by a set of quite different segmentators.

Table 7. Average Q-statistic.

Ensembles	Average Q-Statistic
ERN101(10)	*0.975*
ELOSS101(10)	*0.952*
ELOSSMIX(10)	*0.921*
FH(2)+2×PVT(2)	*0.925*

Chapter 4

Conclusions

In computer vision, we called semantic segmentation the task that involves the classification of each pixel in an image.

This is a very important task in in several fields, e.g. in autonomous vehicles, it allows the identification of objects surrounding the vehicle; in medical diagnosis, it improves the ability of early detecting dangerous pathologies and thus to mitigate the risk of serious consequences.

Here we obtain state-of-the-art performances proposing different ensemble of segmentation approaches. We have tested:

- Different loss functions;
- Different data augmentation approaches;
- Different network topologies, i.e. convolutional neural networks and transformer (namely DeepLabV3+, HarDNet-MSEG, and Pyramid Vision Transformers);

Finally, the ensemble is combined by sum rule.

Our proposed ensemble has been tested, providing state-of-the-art results, in five benchmark datasets: polyp detection, skin detection, leukocytes recognition, environmental microorganism detection, and butterfly recognition.

As future work, our aim - through techniques such as pruning, quantization, low-ranking factorization and distillation - is to decrease the complexity of ensembles.

All resources are available online at:

https://github.com/AlbertoFormaggio1/Ensemble-Of-Segmentation.

Chapter 5
References

[1] D. Feng et al., “Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 3, pp. 1341–1360, 2020.

[2] P. Brandao et al., “Towards a computed-aided diagnosis system in colonoscopy: automatic polyp segmentation using convolution neural networks,” J. Med. Robot. Res., vol. 3, no. 02, p. 1840002, 2018.

[3] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” 2015, doi: 10.1109/ICCV.2015.178.

[4] L. Li, “Deep residual autoencoder with multiscaling for semantic segmentation of land-use images,” Remote Sens., vol. 11, no. 18, 2019, doi: 10.3390/rs11182142.

[5] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” 2018, doi: 10.1007/978-3-030-01234-2_49.

[6] C.-H. Huang, H.-Y. Wu, and Y.-L. Lin, “HarDNet-MSEG: A Simple Encoder-Decoder Polyp Segmentation Neural Network that Achieves over 0.9 Mean Dice and 86 FPS,” Jan. 2021, Accessed: Mar. 30, 2021. [Online]. Available: http://arxiv.org/abs/2101.07172.

[7] B. Dong, W. Wang, J. Li, and D.-P. Fan, “Polyp-PVT: Polyp Segmentation with Pyramid Vision Transformers,” Aug. 2021, doi: 10.48550/arxiv.2108.06932.

[8] D. Jha et al., “Kvasir-SEG: A Segmented Polyp Dataset,” 2020, doi: 10.1007/978-3-030-37734-2_37.

[9] Y. Zhang, H. Liu, and Q. Hu, “TransFuse: Fusing Transformers and CNNs for Medical Image Segmentation,” Feb. 2021, Accessed: Sep. 28, 2021. [Online]. Available: http://arxiv.org/abs/2102.08005.

[10] T. Kim, H. Lee, and D. Kim, “UACANet: Uncertainty Augmented Context Attention for Polyp Segmentation,” Jul. 2021, doi: 10.1145/3474085.3475375.

[11] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, “Image Segmentation Using Deep Learning: A Survey,” IEEE Trans. Pattern Anal. Mach. Intell., 2021, doi: 10.1109/TPAMI.2021.3059968.

[12] A. Lumini and L. Nanni, “Fair comparison of skin detection approaches on publicly available datasets,” Expert Systems with Applications. 2020, doi: 10.1016/j.eswa.2020.113677.

[13] K. Roy, A. Mohanty, and R. R. Sahay, “Deep Learning Based Hand Detection in Cluttered Environment Using Skin Segmentation,” in 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 640–649, doi: 10.1109/ICCVW.2017.81.

[14] M. Arsalan, D. S. Kim, M. Owais, and K. R. Park, “OR-Skip-Net: Outer residual skip network for skin segmentation in non-ideal situations,” Expert Syst. Appl., vol. 141, p. 112922, Mar. 2020, doi: 10.1016/J.ESWA.2019.112922.

[15] S. Shahriar et al., “Real-time american sign language recognition using skin segmentation and image category classification with convolutional neural network and deep learning,” in TENCON 2018-2018 IEEE Region 10 Conference, 2018, pp. 1168–1171.

[16] M. R. Reena and P. M. Ameer, “Localization and recognition of leukocytes in peripheral blood: A deep learning approach,” Comput. Biol. Med., vol. 126, 2020, doi: 10.1016/j.compbiomed.2020.104034.

[17] Y. Liu, F. Cao, J. Zhao, and J. Chu, “Segmentation of White Blood Cells Image Using Adaptive Location and Iteration,” IEEE J. Biomed. Heal. Informatics, vol. 21, no. 6, 2017, doi: 10.1109/JBHI.2016.2623421.

[18] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., 2017, doi: 10.1109/TPAMI.2016.2572683.

[19] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., 2017, doi: 10.1109/TPAMI.2016.2644615.

[20] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” IEEE Trans. Pattern Anal. Mach. Intell., 2018, doi: 10.1109/TPAMI.2017.2699184.

[21] S. Zheng et al., “Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers,” Dec. 2020, Accessed: Sep. 28, 2021. [Online]. Available: https://arxiv.org/abs/2012.15840.

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[23] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” Artif. Intell. Rev., 2020, doi: 10.1007/s10462-020-09825-6.

[24] L. Nanni, D. Cuza, A. Lumini, A. Loreggia, and S. Brahnam, “Deep ensembles in bioimage segmentation,” Dec. 2021, doi: 10.48550/arxiv.2112.12955.

[25] S. Jadon, “A survey of loss functions for semantic segmentation,” 2020, doi: 10.1109/CIBCB48159.2020.9277638.

[26] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations,” 2017, doi: 10.1007/978-3-319-67558-9_28.

[27] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, “Tversky loss function for image segmentation using 3D fully convolutional deep networks,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2017, vol. 10541 LNCS, doi: 10.1007/978-3-319-67389-9_44.

[28] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 2, 2020, doi: 10.1109/TPAMI.2018.2858826.

[29] N. Abraham and N. M. Khan, “A novel focal tversky loss function with improved attention u-net for lesion segmentation,” in Proceedings - International Symposium on Biomedical Imaging, 2019, vol. 2019-April, doi: 10.1109/ISBI.2019.8759329.

[30] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand, “Basnet: Boundary-aware salient object detection,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019, vol. 2019-June, doi: 10.1109/CVPR.2019.00766.

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Trans. Image Process., vol. 13, no. 4, 2004, doi: 10.1109/TIP.2003.819861.

[32] Y. S. Aurelio, G. M. de Almeida, C. L. de Castro, and A. P. Braga, “Learning from Imbalanced Data Sets with Weighted Cross-Entropy Function,” Neural Process. Lett., vol. 50, no. 2, 2019, doi: 10.1007/s11063-018-09977-1.

[33] M. A. Rahman and Y. Wang, “Optimizing intersection-over-union in deep neural networks for image segmentation,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 10072 LNCS, doi: 10.1007/978-3-319-50835-1_22.

[34] Y.-J. Cho, “Weighted Intersection over Union (wIoU): A New Evaluation Metric for Image Segmentation,” Jul. 2021, doi: 10.48550/arxiv.2107.09858.

[35] D. Yang, H. Roth, X. Wang, Z. Xu, A. Myronenko, and D. Xu, “Enhancing Foreground Boundaries for Medical Image Segmentation,” May 2020, doi: 10.48550/arxiv.2005.14355.

[36] Z. Chen, H. Zhou, J. Lai, L. Yang, and X. Xie, “Contour-Aware Loss: Boundary-Aware Learning for Salient Object Segmentation,” IEEE Trans. Image Process., vol. 30, 2021, doi: 10.1109/TIP.2020.3037536.

[37] D. Jha et al., “Real-time polyp detection, localisation and segmentation in colonoscopy using deep learning,” arXiv. 2020.

[38] J. Bernal, J. Sánchez, and F. Vilariño, “Towards automatic polyp detection with a polyp appearance model,” 2012, doi: 10.1016/j.patcog.2012.03.002.

[39] D. Vázquez et al., “A Benchmark for Endoluminal Scene Segmentation of Colonoscopy Images,” J. Healthc. Eng., vol. 2017, 2017, doi: 10.1155/2017/4037190.

[40] J. Silva, A. Histace, O. Romain, X. Dray, and B. Granado, “Toward embedded detection of polyps in WCE images for early diagnosis of colorectal cancer,” Int. J. Comput. Assist. Radiol. Surg., 2014, doi: 10.1007/s11548-013-0926-3.

[41] J. Bernal, F. J. Sánchez, G. Fernández-Esparrach, D. Gil, C. Rodríguez, and F. Vilariño, “WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians,” Comput. Med. Imaging Graph., 2015, doi: 10.1016/j.compmedimag.2015.02.007.

[42] S. L. Phung, A. Bouzerdoum, and D. Chai, “Skin segmentation using color pixel classification: Analysis and comparison,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 1, pp. 148–154, 2005, doi: 10.1109/TPAMI.2005.17.

[43] W. R. Tan, C. S. Chan, P. Yogarajah, and J. Condell, “A Fusion Approach for Efficient Human Skin Detection,” Ind. Informatics, IEEE Trans., vol. 8, no. 1, pp. 138–147, 2012, doi: 10.1109/TII.2011.2172451.

[44] L. Huang, T. Xia, Y. Zhang, and S. Lin, “Human skin detection in images by MSER analysis,” 18th IEEE Int. Conf. Image Process., pp. 1257–1260, 2011, doi: 10.1109/ICIP.2011.6115661.

[45] J. Ruiz-Del-Solar and R. Verschae, “Skin detection using neighborhood information,” in Proceedings - Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, pp. 463–468, doi: 10.1109/AFGR.2004.1301576.

[46] M. J. Jones and J. M. Rehg, “Statistical color models with application to skin detection,” Int. J. Comput. Vis., vol. 46, no. 1, pp. 81–96, 2002, doi: 10.1023/A:1013200319198.

[47] J. P. B. Casati, D. R. Moraes, and E. L. L. Rodrigues, “SFA: A human skin image database based on FERET and AR facial images,” 2013.

[48] M. Kawulok, J. Kawulok, J. Nalepa, and B. Smolka, “Self-adaptive algorithm for segmenting skin regions,” EURASIP J. Adv. Signal Process., no. 1, pp. 1–22, 2014, doi: 10.1186/1687-6180-2014-170.

[49] S. J. Schmugge, S. Jayaram, M. C. Shin, and L. V. Tsap, “Objective evaluation of approaches of skin detection using ROC analysis,” Comput. Vis. Image Underst., vol. 108, no. 1–2, pp. 41–51, 2007, doi: 10.1016/j.cviu.2006.10.009.

[50] J. C. Sanmiguel and S. Suja, “Skin detection by dual maximization of detectors agreement for video monitoring,” Pattern Recognit. Lett., vol. 34, no. 16, pp. 2102–2109, 2013, doi: 10.1016/j.patrec.2013.07.016.

[51] A. S. Abdallah, M. A. El-Nasr, and A. L. Abbott, “A new color image database for benchmarking of automatic face detection and human skin segmentation techniques,” in Proceedings of World Academy of Science, Engineering and Technology, 2007, vol. 20, pp. 353–357.

[52] J. Wang, K. Markert, and M. Everingham, “Learning models for object recognition from natural language descriptions,” 2009, doi: 10.5244/C.23.2.

[53] P. Zhao et al., “EMDS-6: Environmental Microorganism Image Dataset Sixth Version for Image Denoising, Segmentation, Feature Extraction, Classification, and Detection Method Evaluation.,” Front. Microbiol., vol. 13, p. 829027, 2022, doi: 10.3389/fmicb.2022.829027.

[54] J. Chen et al., “TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation,” Feb. 2021, Accessed: Sep. 28, 2021. [Online]. Available: https://arxiv.org/abs/2102.04306.

[55] J. Wei, Y. Hu, R. Zhang, Z. Li, S. K. Zhou, and S. Cui, “Shallow Attention Network for Polyp Segmentation,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021, vol. 12901 LNCS, doi: 10.1007/978-3-030-87193-2_66.

[56] X. Zhao, L. Zhang, and H. Lu, “Automatic Polyp Segmentation via Multi-scale Subtraction Network,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021, vol. 12901 LNCS, doi: 10.1007/978-3-030-87193-2_12.

[57] K.-B. Park and J. Y. Lee, “SwinE-Net: hybrid deep learning approach to novel polyp segmentation using convolutional neural network and Swin Transformer,” J. Comput. Des. Eng., vol. 9, no. 2, pp. 616–632, Apr. 2022, doi: 10.1093/jcde/qwac018.

[58] P. Song, J. Li, and H. Fan, “Attention based multi-scale parallel network for polyp segmentation,” Comput. Biol. Med., vol. 146, p. 105476, Jul. 2022, doi: 10.1016/J.COMPBIOMED.2022.105476.

[59] I. Filali, B. Achour, M. Belkadi, and M. Lalam, “Graph ranking based butterfly segmentation in ecological images,” Ecol. Inform., vol. 68, 2022, doi: 10.1016/j.ecoinf.2022.101553.

[60] H. Tang, B. Wang, and X. Chen, “Deep learning techniques for automatic butterfly segmentation in ecological images,” Comput. Electron. Agric., vol. 178, 2020, doi: 10.1016/j.compag.2020.105739.

[61] P. Pudil, J. Novovičová, and J. Kittler, “Floating search methods in feature selection,” Pattern Recognit. Lett., 1994, doi: 10.1016/0167-8655(94)90127-9.

[62] L. I. Kuncheva and C. J. Whitaker, “Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy,” Mach. Learn., 2003, doi: 10.1023/A:102285900300

Acknowledgements

Un sentito ringraziamento va a tutte le persone che mi hanno aiutato a raggiungere questo traguardo, nello specifico:

- Al professor Loris Nanni a cui sono riconoscente per avermi dato l'opportunità di lavorare con lui a questa ricerca e per essere sempre stato disponibile ad aiutarmi
- Ai miei genitori, i quali hanno sempre creduto in me e mi hanno supportato sin dall'infanzia: avete sempre voluto il meglio per me e mi avete sempre spinto in questa direzione, apprezzo davvero tutto ciò che avete fatto affinché io potessi raggiungere questo importante traguardo della mia vita.
- Alle mie nonne, che durante questi tre anni universitari hanno sempre seguito il mio percorso accademico.
- Ai miei amici e compagni di corso con cui ho vissuto dei momenti speciali che mi hanno aiutato anche nei periodi più difficili.
- Ad Andrea, il migliore amico che si possa desiderare e la persona su cui so di poter sempre contare nonostante tutto.