End-to-end Ultrametric Learning for Hierarchical Segmentation

Raphael Lapertot, Giovanni Chierchia, Benjamin Perret

End-to-end Ultrametric Learning for Hierarchical Segmentation *

Raphael Lapertot[1 [0009−0004−1208−8115]], Giovanni Chierchia[1 [0000−0001−5899−689X]], and Benjamin Perret[1 [0000−0003−0933−8342]]

LIGM, Univ Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée, France

Abstract. Hierarchical image segmentation aims to capture the structure of objects of different sizes at different scales and helps to understand the scene. With the success of neural networks for image segmentation and the recent emergence of object and part segmentation datasets, the task of supervised learning of segmentation hierarchies naturally arises. In a previous work, we proposed a differentiable ultrametric layer that transforms any dissimilarity measure into an ultrametric distance equivalent to a hierarchical segmentation. In this paper, we study several loss functions for end-to-end learning of a neural network model predicting hierarchical segmentations. In particular, we propose a generalization of the Rand index for hierarchical segmentation and propose exact and approximate algorithms to compute it. We introduce new metrics to compare hierarchical segmentations, and we demonstrate the suitability of the proposed pipeline with several possible loss function combinations on a simulated hierarchical dataset.

Keywords: Image segmentation · Hierarchy · Ultrametric

1 Introduction

Image segmentation is the process of dividing an image into distinct regions that highlight relevant structures. One critical factor in this process is the selection of an appropriate scale, since it affects the level of detail visible in an image. Such difficulty can be avoided by using a hierarchical framework that generates consistent segmentations across multiple scales. This approach allows one to defer the decision regarding the scale until after the segmentation is complete.

With the success of neural networks in flat image segmentation [7,23,25,21] and the recent emergence of hierarchical segmentation datasets [4,9,19,14,8], the task of end-to-end supervised learning of hierarchical segmentation naturally arises. Notable efforts have been made to produce high quality hierarchical image segmentations, however, to the best of our knowledge, none of them propose loss functions for end-to-end supervised learning of hierarchical image segmentation. Some prior research has performed end-to-end supervised learning of

* This work is supported by the French ANR grant ANR-20-CE23-0019, and was granted access to the HPC resources of IDRIS under the allocation 2023-AD011013101R1 made by GENCI.

flat image segmentations [24,1,16], while other studies have produced hierarchical segmentations, but not in an end-to-end fashion, using flat segmentation ground-truths [20,18,13]. HieraSeg [12] propose a pixel approach for hierarchical semantic segmentation, for which the hierarchy is on the semantic labels. Loss functions play a major role in the process as they guide the model towards meaningful segmentation hierarchies through the optimization process. Using an end-to-end pipeline has also shown to be advantageous.

This paper extends our previous works which proposed a differentiable ultrametric layer [5], and a pipeline for the supervised learning of ultrametrics [11]. We explore various loss functions including an adaptation of the Rand Index, a well-established metric for evaluating segmentation quality, to the hierarchical context for end-to-end optimization. We also introduce quantitative metrics to assess well-ordered hierarchical segmentations. Our main contributions are new loss functions for the supervised learning of hierarchical segmentation, corresponding algorithms for approximate or exact computation, and quantitative metrics for hierarchical segmentation assessments.

2 Model

Our primary objective is to establish an end-to-end supervised learning framework for hierarchical segmentation. Given the success of deep learning methodologies in the domain of image segmentation, it is natural to explore their applicability to this particular context. We frame the problem as a regression task operating on the edges of the 4-adjacency graph (or grid) $\mathcal{G}$ of the image, with pixels as vertices and edges linking each neighbor pixels.

2.1 Hierarchical segmentation using graphs

Flat segmentation is often approached from a region-based perspective. From this point of view, a segmentation S associates each pixel v of the image with an associated region label s_v. It can also be seen from a boundary-based perspective. Let us consider the 4-adjacency graph of an image. Considering a segmentation S, the *cut* ϕ_S indicates whether two pixels belong to the same region in S or not: $\phi_S(v, v') = 1$ if $s_v \neq s_{v'}$ else 0 with v and v' two pixels.

Hierarchical segmentation can also be seen from a region point of view and a boundary point of view. In the first case, it is represented as a sequence of ordered segmentations, the next one being a refinement of the previous one. From the boundary perspective, a hierarchical image segmentation is represented as an ultrametric (dissimilarity) grid $(\mathcal{G}, w)$ (also called saliency map or ultrametric contour map), that is, a 4-adjacency graph $\mathcal{G}$ whose edges $e \in E$ are weighted by a dissimilarity value $w(e)$ that satisfies the ultrametric constraint $(\forall C \in \mathcal{C}, \forall e \in C, w(e) \leq \max_{e' \in C \setminus \{e\}} w(e')$, with $\mathcal{C}$ the set of cycles of $\mathcal{G})$.

In general, a dissimilarity grid $(\mathcal{G}, w)$ is not an ultrametric grid, but a simple way to transform it into an ultrametric grid is to compute its *subdominant ultrametric* $(\mathcal{G}, u_w)$, *i.e.*, the largest ultrametric grid which is smaller or equal

than $(\mathcal{G}, w)$. In image analysis, computing the subdominant ultrametric of a dissimilarity grid can be seen as *removing borders with holes*. The subdominant ultrametric of a dissimilarity grid can also be defined thanks to the notion of *min-max paths*, *i.e.* paths that minimizes the maximal value along the path. More formally, we denote $\mathcal{P}_{v,v'}$ the set of paths from v to v' in a dissimilarity grid $(\mathcal{G}, w)$. The min-max path $P^*_{v,v'}$ from the pixel v to the pixel v' is

$$P^*_{v,v'} = \underset{P \in \mathcal{P}_{v,v'}}{\arg\min} \max_{e \in P} w(e). \tag{1}$$

The maximal edge of the min-max path is called the *min-max edge*, and will be denoted $mm_w(v, v')$. Finally, the subdominant ultrametric $d_w(v, v')$ of $(\mathcal{G}, w)$ on the edge $\{v, v'\}$ is equal to the weight of the min-max edge between v and v':

$$d_w(v, v') = w(mm_w(v, v')). \tag{2}$$

In this paper, we will consider hierarchies represented as ultrametric grid.

2.2 Ultrametric network

We propose the design of a neural network that predicts an ultrametric grid from an input image. The pipeline is shown in Figure 1 and unfolds as follows.

- The input image is given to a U-Net neural network. The network outputs a transformed image of identical dimensions, preserving spatial relationships.
- The output is transformed into edge-weights that correspond to the 4-adjacency graph of the original image.
- A sigmoid activation function binds these edge-weights within the range of 0 (no contour) to 1 (strong contour).
- An ultrametric layer converts the graph into an ultrametric graph, while preserving the differentiability of the computation [5].

In particular, the U-Net architecture shows versatility by accommodating different convolutional or transformer-based neural networks, ensuring adaptability to upcoming innovations. Our approach converts the U-Net's output in the pixel domain to edge-weights by taking the average weight of neighboring pixels.

3 Loss functions

In this section, we introduce differentiable loss functions for comparing two ultrametric grids, one of which is predicted by the proposed ultrametric network. We give algorithms to compute them both approximately and exactly. These loss functions will be used for training the proposed ultrametric network.

In the following, we assume that any ground-truth ultrametric grid w is composed of a few different levels $\Lambda(w)$. For example, a ground-truth in an *objects-and-parts* dataset typically has three levels: the level 0 denotes the absence of frontier, the level 1 delineates objects, and the level 2 defines the parts of objects.

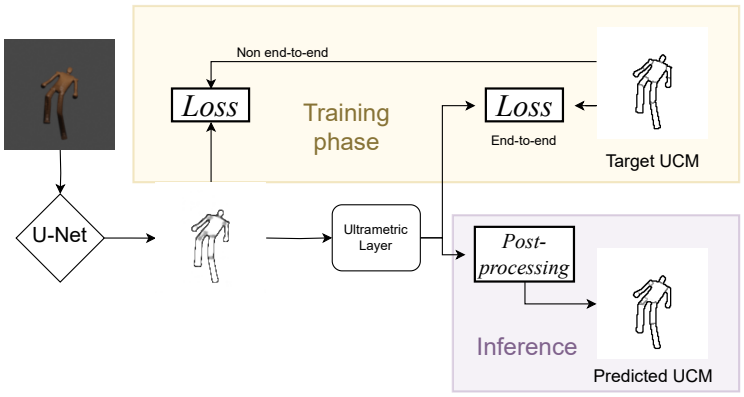


Fig. 1: Our proposed pipeline with optimization either end-to-end or not. In this figure, the U-Net outputs edge-weights (seen as a dissimilarity measure) of the 4-adjacency graph built on the input image. The loss function is computed either before or after (or both) the ultrametric layer that transforms the dissimilarity grid into an ultrametric one, which is equivalent to a hierarchical segmentation.

3.1 Quadratic Error

A simple approach is to consider the L2 distance between the ground-truth ultrametric grid (w) and the predicted edge-weights ($\hat{w}$), while mitigating the imbalance between the number of edges in the different levels of the ground-truth. This can be formulated as

$$L2(\hat{w}, w) = \frac{1}{|E|} \sum_{\lambda \in \Lambda(w)} \beta_\lambda \sum_{e \in E_\lambda} (\hat{w}(e) - w(e))^2, \tag{3}$$

where E denotes the set of edges within the 4-adjacency graph corresponding to the input image, E_λ represents the set of edges with a ground-truth value of λ, and $\beta_\lambda = 1 - \frac{|E_\lambda|}{|E|}$ serves as a class-balancing weight. The latter term is essential as there are typically a lot more non-edges than edges; when this term is not included, the non-edges are naturally privileged, and actual edges are harder to learn and detect. While this approach seems natural, it is sensitive to small geometric transformations such as translations, which is undesirable since the quality of a segmentation should not decrease significantly with small spatial variations. This loss can be applied after the ultrametric layer, offering an end-to-end learning pipeline. It can also be applied before the ultrametric layer, which can serve as a baseline even if the training is no longer end-to-end.

3.2 Hierarchical Rand Index

From a region-based perspective, the Rand Index for two segmentations S and $\hat{S}$ can be defined as follows

$$1 - RI(\hat{S}, S) = \binom{N}{2}^{-1} \sum_{v < v'} \left| \delta(s_v, s_{v'}) - \delta(\hat{s_v}, \hat{s_{v'}}) \right|. \tag{4}$$

In this equation, N is the number of pixels, with $\binom{N}{2}^{-1}$ used for normalization, v and v' are pixels, and $\delta(x, y)$ if the indicator function which is 0 if $x = y$ and 1 of $x \neq y$. Overall, the Rand index measures the proportion of pixel pairs for which the two segmentations disagree. This measure increases when regions are split or merged incorrectly, while being robust to small geometric transformations. From a boundary-based perspective, the Rand Index can be expressed using the cuts $\phi_{\hat{S}}$ and ϕ_S similarly. This is from this boundary-based perspective that we relax the Rand Index to hierarchical segmentation. We define the Hierarchical Rand Index (HRI) of two dissimilarity grids w and $\hat{w}$ on $\mathcal{G}$ as

$$HRI(\hat{w}, w) = \binom{N}{2}^{-1} \sum_{v<v'} \beta_{d_w(v,v')} \left(d_w(v, v') - d_{\hat{w}}(v, v')\right)^2 . \tag{5}$$

It computes the mean ultrametric distance error for every pixel pair of the graph. This measure keeps the same benefits as the Rand index, that is, it focuses on the incorrect merging and splitting of hierarchical regions. We use the same class-balancing weights β_λ as described previously.

3.3 Naive algorithm for HRI

Let us focus on the computation of the HRI. Firstly, in an optimization framework, the loss function needs to be differentiable on the edges of the graph. To calculate the ultrametric distance between pairs of pixels, we exploit an interesting property of minimal spanning trees (or *MST* for short): each path in an MST of a graph is a min-max path. A binary partition tree (*BPT*) associated with the MST can be built during the Kruskal algorithm [15] which enables to efficiently find min-max edge between any pixel pair by querying their lowest common ancestor in this BPT, which can be done efficiently with a linear time preprocessing of the tree [3]. A naive algorithm for computing the HRI is detailed in Algorithm 1. This naive algorithm has a quadratic runtime w.r.t. the number of vertices as it browses each pixel pair.

A first approach to alleviate this problem is to approximate the HRI by considering a subset of M pixel pairs. This reduces the time complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(M + N \log N)$ but we only get an approximation of the correct result. In the context of a stochastic gradient descent algorithm, such kind of approximation in the loss function might be acceptable.

3.4 Optimized algorithm for HRI

In this section, we propose a more efficient algorithm to compute the HRI. This algorithm leverages on the properties of min-max paths and the BPT. The idea is as follows: when computing the MST of a graph and the corresponding BPT using Kruskal's algorithm, when an edge is added to the BPT, it merges two subtrees. The amount of pixels in a tree will be called its *area*. The number of pixel pairs for which the added edge is the min-max edge can be deduced by

Algorithm 1 Naive hierarchical Rand index

procedure NAIVE-HRI($\mathcal{G}, \hat{w}, w$) ▷ $\mathcal{O}(N^2)$
  Compute $BPT(\mathcal{G}, \hat{w})$ ▷ $\mathcal{O}(N \log N)$
  Compute $BPT(\mathcal{G}, w)$ ▷ $\mathcal{O}(N \log N)$
  Compute the class-balancing weight: $\forall \lambda \in \Lambda(w), \beta_\lambda = 1 - \frac{|E_\lambda|}{|E|}$ ▷ $\mathcal{O}(N)$
  for $(v, v') \in V^2$ with $v < v'$ **do** ▷ $\mathcal{O}(N^2)$
    Find the predicted min-max edge $mm_{\hat{w}}(v, v')$ using $BPT(\mathcal{G}, \hat{w})$ ▷ $\mathcal{O}(1)$
    Get its dissimilarity value: $d_{\hat{w}}(v, v') \leftarrow \hat{w}(mm_{\hat{w}}(v, v'))$ ▷ $\mathcal{O}(1)$
    Find the ground-truth min-max edge $mm_w(v, v')$ using $BPT(\mathcal{G}, w)$ ▷ $\mathcal{O}(1)$
    Get its dissimilarity value: $d_w(v, v') \leftarrow w(mm_w(v, v'))$ ▷ $\mathcal{O}(1)$
    Compute error: $HRI(\hat{w}, w)_{v,v'} = \beta_{d_w(v,v')} \left(d_w(v, v') - d_{\hat{w}}(v, v')\right)^2$ ▷ $\mathcal{O}(1)$
  end for
  Return the mean HRI error
end procedure

looking at the areas of the trees it merges. Formally, for each edge $e \in MST(\mathcal{G}, \hat{w})$ and each ground-truth level $\lambda \in \Lambda(w)$, we define the area $A(e, \lambda)$ at e for the level λ as the number of pixel pairs whose min-max edge in $(\mathcal{G}, \hat{w})$ is e and whose ultrametric distance in the ground truth is λ:

$$A(e, \lambda) = \left|\{(v, v') \in V^2 \mid d_w(v, v') = \lambda, e = mm_{\hat{w}}(v, v')\}\right|. \tag{6}$$

The HRI can then be rewritten as follows

$$HRI(\hat{w}, w) = \binom{N}{2}^{-1} \sum_{\lambda \in \Lambda(w)} \sum_{e \in MST(\mathcal{G}, \hat{w})} A(e, \lambda)(\lambda - \hat{w}(e))^2. \tag{7}$$

In this new formulation which is equivalent to Equation 5, we have replaced the sum over all pixel pairs by a double sum over the ground truth levels, which is small, and the MST edges, which is in the order of N the number of pixels.

Then, let's see how to calculate $A(e, \lambda)$. Assume that the finest ground-truth partition if composed of k regions labeled $\{1, ..., k\}$. We define the ultrametric distance between any two of these regions as

$$U(i, j) = d_w(v, v'). \tag{8}$$

where v (resp v') is a pixel of the region i (resp. j); note that the choice of v and v' does not matter as they are all at the same ultrametric distance. Let e be an edge of $MST(\mathcal{G}, \hat{w})$. This edge corresponds to a node n in the associated BPT. The number of pixels of label i contained in the subtree rooted at the node n is denoted $R(n, i)$. Let c_1^e and c_2^e be respectively the left and right children of the node of the BPT associated to the edge e. Note that, as e merges its left and right subtrees in the BPT, e is the min-max edge between any pixel in the subtrees rooted in c_1^e and in c_2^e. We can observe that, for any MST edge e and for any two different regions i and j, there are $R(c_1^e, i)R(c_2^e, j) + R(c_1^e, j)R(c_2^e, i)$ pairs of pixels from the regions i and j in the ground-truth between the subtrees rooted in c_1^e and c_2^e. Similarly, there are $R(c_1^e, i)R(c_2^e, i)$ pairs of pixels of the same

Algorithm 2 Optimized hierarchical Rand index

procedure OPTIMIZED-HRI($\mathcal{G}, \hat{w}, w$) ▷ $\mathcal{O}(N(\log N + k^2 + |\Lambda(w)|))$
  Compute $BPT(\mathcal{G}, \hat{w})$ ▷ $\mathcal{O}(N \log N)$
  Compute $BPT(\mathcal{G}, w)$ ▷ $\mathcal{O}(N \log N)$
  Compute the ultrametric distance of regions of the finest partition: U ▷ $\mathcal{O}(k^2)$
  Compute areas of regions per edge of $MST(\mathcal{G}, \hat{w})$: R ▷ $\mathcal{O}(Nk)$
  Compute areas of regions pair per edge of $MST(\mathcal{G}, \hat{w})$: B ▷ $\mathcal{O}(Nk^2)$
  Compute areas per ultrametric value and edge of $MST(\mathcal{G}, \hat{w})$: A ▷ $\mathcal{O}(N|\Lambda(w)|)$
  Compute $HRI = \sum_{\lambda \in \Lambda(w)} \sum_{e \in MST(\mathcal{G}, \hat{w})} A(e, \lambda)(\lambda - \hat{w}(e))^2$ ▷ $\mathcal{O}(N|\Lambda(w)|)$
  Return HRI error ▷ $\mathcal{O}(1)$
end procedure

region i in the ground-truth between the subtrees rooted in c_1^e and c_2^e. Thus, the number of pixels pairs of region i and j, whose min-max edge is e, reads

$$B(e, i, j) = \begin{cases} R(c_1^e, i)R(c_2^e, j) + R(c_1^e, j)R(c_2^e, i) & \text{if } i \neq j \\ R(c_1^e, i)R(c_2^e, i) & \text{if } i = j. \end{cases} \tag{9}$$

Then, the area $A(e, \lambda)$ indicating for each edge e of the MST and for each ultrametric value $\lambda \in \Lambda(w)$, how many pixel pairs of ultrametric distance λ in the ground-truth have the min-max edge e in $\hat{w}$, can be rewritten as:

$$A(e, \lambda) = \sum_{i=1}^{k} \sum_{j=i}^{k} \Big(1 - \delta\big(\lambda, U(i, j)\big)\Big) B(e, i, j). \tag{10}$$

In practice, R is a $(N-1) \times k$ matrix and B is a $(N-1) \times k^2$ matrix: they can be seen as attributes mapping a k (resp. k^2) vector to any internal node of the BPT. Both can be computed by browsing the BPT from the leaves to the root.

The complete HRI can be computed by Algorithm 2 with a runtime complexity of $\mathcal{O}\big(N(\log N + k^2 + |\Lambda(w)|)\big)$. The fewer regions in the ground-truth, the faster the algorithm. Both Algorithms 1 and 2 yield the same result.

4 Experiments

We now evaluate the proposed loss functions. Specifically, we analyze the effectiveness of these loss functions for training a neural network in the context of hierarchical image segmentation.

4.1 Dataset

We use a custom-made *Humanoid* Dataset, which is a toy dataset consisting of a virtual humanoid moving in front of 6 cameras with different angles for 120 frames, resulting in 720 images of size 128x128. The ground-truths consists of Ultrametric Contour Maps (UCM) [2,6,11], where interpixels are added to the

original image to represent the edges, resulting in an image twice as large as the input image. These UCMs have object-edges drawing the external contours, and part-edges drawing the contours of the parts (fore-arms, face, ...), with respectively value 1 and 0.5. When there are no edges, the value is 0. This dataset was generated using Blender. In our experiments, 4 cameras are used for training, 1 camera is used for validation, and the remaining camera is used for testing. Three samples of the Humanoid Dataset are displayed in Figure 2.

4.2 Neural Network & hyper-parameters

We use a ResNet18 pretrained on ImageNet for the backbone of our U-Net. We first freeze the encoder to only train the decoder for 300 epochs with a One Cycle learning rate scheduler [22] with a maximum learning rate of 0.001. We use an Adam optimizer with a weight decay of 0.01. Once this pre-training is done, we unfreeze the encoder and train it for an additional 100 epochs with a One Cycle learning rate scheduler with a maximum learning rate of 0.0002. We use a batch-size of 64 and simple data augmentations such as horizontal flips and slight random brightness, saturation and contrast modifications.

In terms of computation time, training the neural network as described above using four NVIDIA Tesla V100 GPUs, the partial algorithm took around 47 minutes to compute with one pixel pair, approximately 51 minutes with 16384 pixel pairs, whereas the optimized algorithm required approximately 1 hour and 16 minutes. For reference, the control group with the L2 loss prior to the ultrametric layer took around 32 minutes to train. Therefore, in the experiments, we used 16384 random pixel pairs when computing the partial algorithm.

The network predictions are post-processed with an area filter removing 1-pixel regions [17]. The ultrametric grids are then converted to Ultrametric Contour Maps (UCM) [2,6,11], where interpixels are added to the original image to represent the edges, yielding an image twice as large as the input.

4.3 Metrics

For assessing hierarchical segmentation, we build on our previous work [11] using the Level Recovery Fraction (LRF) and the False Discovery Fraction (FDF). Let U_{pred} be a predicted UCM and U_{tar} be the corresponding ground-truth UCM. The threshold of U_{pred} at levels t is denoted by U_{pred_t}.

For any ground-truth level λ and any threshold level t, $LRF(\lambda, t)$ denotes the fraction of ground-edges of level λ which are matched to an edge in U_{pred_t}:

$$LRF(\lambda, t) = \frac{\left|\{e \mid \text{match}_{U_{pred_t}}(e) \text{ and } U_{tar}(e) = \lambda\}\right|}{|\{e \mid U_{tar}(e) = \lambda\}|}, \tag{11}$$

where $\text{match}_{U_{pred_t}}(e)$ is true if the edge e is matched with a corresponding edge in U_{pred_t} using a bipartite graph matching method.

For the False Discovery Fraction, it is however not immediate to associate a false positive edge in the prediction to a specific level of the ground-truth. For

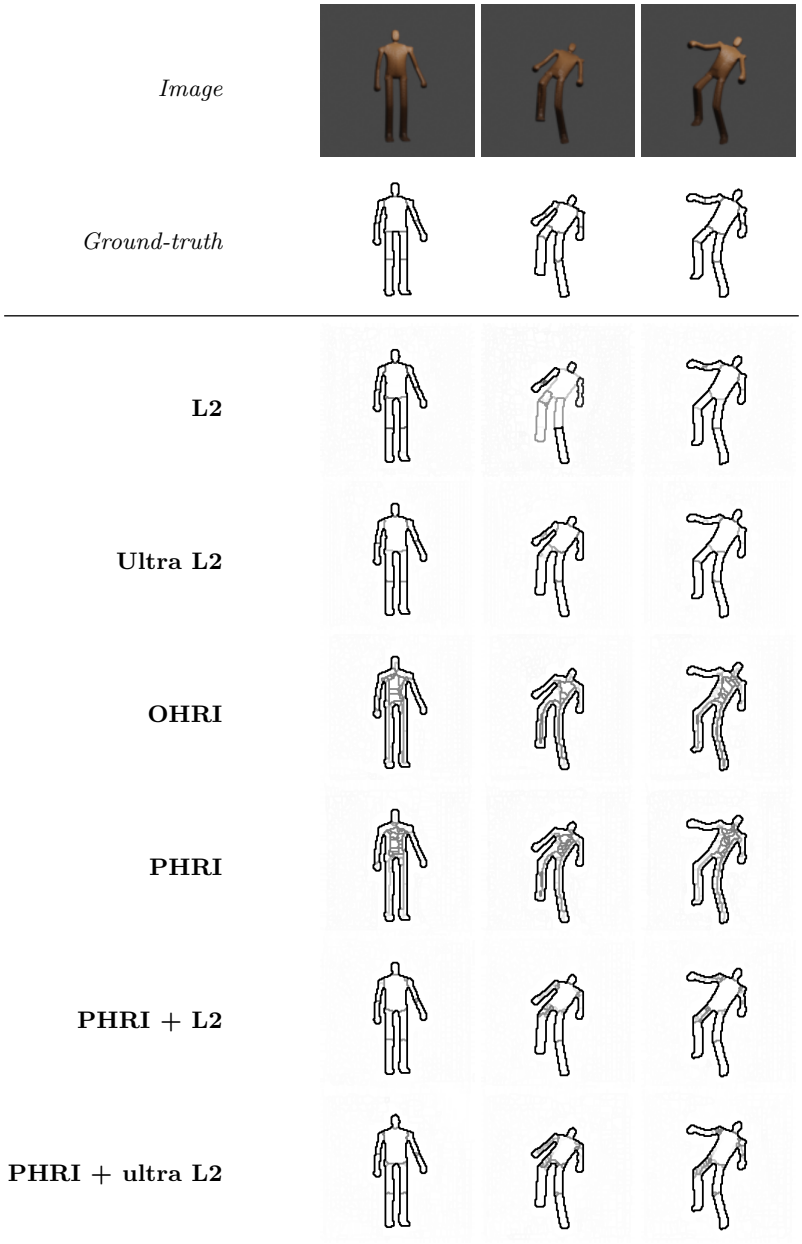


Fig. 2: Predictions of the neural networks on the Humanoid Dataset trained with different loss functions. When training with L2 alone, some contours are opened, resulting in edges that are less salient than expected after post-processing. When training with HRI alone, the object-edges are well detected and closed, but the part-edges are not well located. The other configurations produce closed edges that are well located.

any threshold level t, $FDF(t)$ is then defined as the fraction of prediction edges at the threshold t that cannot be matched with any edge of the ground-truth:

$$FDF(t) = \frac{\left|\{e \mid !\text{match}_{U_{pred_t}}(e) \text{ and } U_{pred_t}(e) = 1\}\right|}{|\{e \mid U_{pred_t}(e) = 1\}|}. \tag{12}$$

In a good prediction, the object boundaries will be recovered at a high threshold (1 in our setting), and the part boundaries will be recovered at a medium threshold (0.5 in our setting), resulting in two step-functions. The FDF, on the other hand, would be 0 for every threshold.

To summarize LRF curves, we propose to calculate the area enclosed between the ideal step curve and the observed curve within the LRF at each level of the hierarchy (LRF_{obj} for object boundaries, and LRF_{part} for part boundaries). Additionally, we compute the mean value of the FDF among the thresholds, giving $mFDF$. Smaller values for these metrics indicate improved performance. All in all, LRF_{obj} and LRF_{part} measure if detected edges are in the right order in the hierarchy, and $mFDF$ gives insights about false detections.

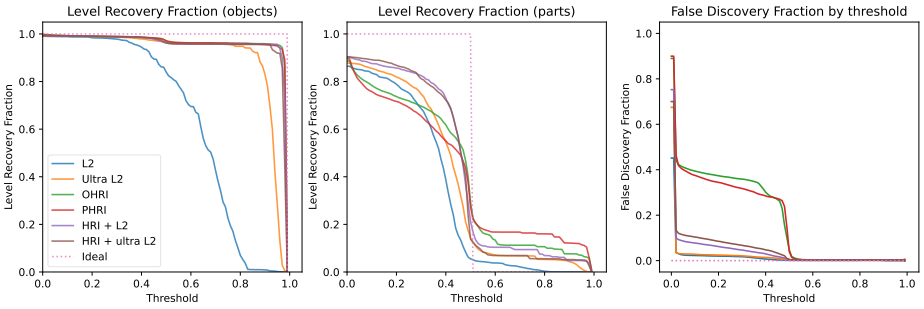
4.4 Results

We address the following problematic: how effective the loss functions are for training a neural network for hierarchical image segmentation? To this end, we train neural networks with various combinations of the proposed loss functions. We automatically combine multiple losses using learnable parameters [10,26]. In the following, *L2* stands for the quadratic error loss applied before the ultrametric layer (not end-to-end) and serves as a control group, *UL2* is the quadratic error loss applied after the ultrametric layer (end-to-end), and *OHRI* and *PHRI* are the hierarchical rand index computed with respectively the optimized algorithm and the partial algorithm (both end-to-end).

The metrics on the test set are shown in Figure 3, and the predictions of those networks are displayed in Figure 2. The Level Recovery Fraction (objects) plot shows the limits of the control group when computing the L2 prior on the ultrametric layer; the blue line representing object boundaries recovery shows that object borders are recovered at a medium threshold instead of a high threshold, resulting in a high LRF_{obj} error. This effect, visible in Figure 2, is mostly due to open borders that are removed when computing the subdominant ultrametric. The aforementioned effect does not happen with the other loss functions.

Part boundaries are recovered at a medium threshold for most loss functions, except when training with HRI alone (green and red line). For the latter, part boundaries are recovered more linearly at a mid-low threshold (instead of the ideal step curve at a mid threshold), and wrongly recovered even at a high threshold, resulting in a high LRF_{part} error. The control group also recovers part boundaries at a lower threshold than other methods.

Finally, the False Discovery Fraction plot shows that training with HRI alone results in a lot of wrong boundaries at a medium to low threshold, mostly because it does not learn where to place the part borders as visible in Figure 2.



	LRF_{obj}	LRF_{part}	$mFDF$
L2	0.354	0.200	0.013
UL2	0.086	0.184	0.018
OHRI	0.031	0.211	0.180
PHRI	0.030	0.250	0.171
PHRI+L2	0.036	0.155	0.034
PHRI+UL2	0.040	0.144	0.044

Fig. 3: Experiment results. The errors are the area between the ideal case (dotted line) and the actual line for the three metrics. Lower errors means better results.

5 Conclusion

In this paper, we presented new loss functions and developed algorithms for both approximate and optimized computation. We demonstrated their effectiveness through a proof-of-concept experiment on a simplified dataset. Additionally, we introduced quantitative metrics for evaluating hierarchical segmentation. Future research will involve applying this method to larger and more complex hierarchical datasets to further evaluate its potential. Additionally, there is a need for further investigation into the selection of pixel pairs for the approximate computation of the HRI. A particularly intriguing and challenging direction for future research is the incorporation of a semantic dimension into the hierarchical segmentation pipeline.

References

1. Al-Huda, Z., Peng, B., Yang, Y., Algburi, R.N.A.: Object scale selection of hierarchical image segmentation with deep seeds. IET Image Processing **15**, 191–205 (2020)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE TPAMI **33**(5), 898–916 (2010)
3. Bender, M.A., Farach-Colton, M.: The lca problem revisited. In: LATIN 2000: Theoretical Informatics. pp. 88–94. Springer (2000)
4. Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., Yuille, A.: Detect what you can: Detecting and representing objects using holistic models and body parts. In: IEEE CVPR (June 2014)
5. Chierchia, G., Perret, B.: Ultrametric fitting by gradient descent. In: NeurIPS. vol. 32. Curran Associates, Inc. (2019)

6. Cousty, J., Najman, L., Kenmochi, Y., Guimarães, S.: Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps. JMIV **60**(4), 479–502 (2018)
7. Elizar, E., Zulkifley, M.A., Muharar, R., Zaman, M.H.M., Mustaza, S.M.: A review on multiscale-deep-learning applications. Sensors **22**(19) (2022)
8. de Geus, D., Meletis, P., Lu, C., Wen, X., Dubbelman, G.: Part-aware panoptic segmentation. In: IEEE CVPR (2021)
9. He, J., Yang, S., Yang, S., Kortylewski, A., Yuan, X., Chen, J., Liu, S., Yang, C., Yuille, A.L.: Partimagenet: A large, high-quality dataset of parts. CoRR (2021)
10. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics (2018)
11. Lapertot, R., Chierchia, G., Perret, B.: Supervised Learning of Hierarchical Image Segmentation. In: CIARP (2023)
12. Li, L., Zhou, T., Wang, W., Li, J., Yang, Y.: Deep hierarchical semantic segmentation. In: IEEE CVPR. pp. 1236–1247 (2022)
13. Maninis, K., Pont-Tuset, J., Arbeláez, P., Gool, L.V.: Convolutional oriented boundaries: From image segmentation to high-level tasks. IEEE TPAMI (2017)
14. Meletis, P., Wen, X., Lu, C., de Geus, D., Dubbelman, G.: Cityscapes-panoptic-parts and pascal-panoptic-parts datasets for scene understanding. CoRR (2020)
15. Najman, L., Cousty, J., Perret, B.: Playing with Kruskal: algorithms for morphological trees in edge-weighted graphs. In: ISMM. LNCS, vol. 7883 (2013)
16. Ôn Vû Ngoc, M., Chen, Y., Boutry, N., Chazalon, J., Carlinet, E., Fabrizio, J., Mallet, C., Géraud, T.: Introducing the Boundary-Aware loss for deep image segmentation. In: BMVC 2021 (2021)
17. Perret, B., Cousty, J., Guimarães, S.J.F., Kenmochi, Y., Najman, L.: Removing non-significant regions in hierarchical clustering and segmentation. PRL **128**, 433–439 (2019)
18. Pont-Tuset, J., Arbeláez, P., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping for image segmentation and object proposal generation. In: arXiv:1503.00848 (March 2015)
19. Ramanathan, V., Kalia, A., Petrovic, V., Wen, Y., Zheng, B., Guo, B., Wang, R., Marquez, A., Kovvuri, R., Kadian, A., Mousavi, A., Song, Y., Dubey, A., Mahajan, D.: Paco: Parts and attributes of common objects (2023)
20. Ren, Z., Shakhnarovich, G.: Image segmentation by cascaded region agglomeration. In: IEEE CVPR. pp. 2011–2018 (2013)
21. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI 2015. pp. 234–241. Springer (2015)
22. Smith, L.N., Topin, N.: Super-convergence: Very fast training of residual networks using large learning rates. CoRR (2017)
23. Thisanke, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R., Herath, D.: Semantic segmentation using vision transformers: A survey (2023)
24. Wolf, S., Schott, L., Köthe, U., Hamprecht, F.: Learned watershed: End-to-end learning of seeded segmentation (2017)
25. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers (2021)
26. Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W.: FairMOT: On the fairness of detection and re-identification in multiple object tracking. IJCV **129**(11), 3069–3087 (sep 2021)