Enhancing Generic Segmentation with Learned Region Representations

Or Isaacs* Oran Shayer* Michael Lindenbaum
Computer Science, Technion - Israel Institute of Technology
orisaacs@gmail.com oran.sh@gmail.com mic@cs.technion.ac.il

Abstract

Current successful approaches for generic (non-semantic) segmentation rely mostly on edge detection and have leveraged the strengths of deep learning mainly by improving the edge detection stage in the algorithmic pipeline. This is in contrast to semantic and instance segmentation, where DNNs are applied directly to generate pixel-wise segment representations. We propose a new method for learning a pixel-wise representation that reflects segment relatedness. This representation is combined with an edge map to yield a new segmentation algorithm. We show that the representations themselves achieve state-of-the-art segment similarity scores. Moreover, the proposed combined segmentation algorithm provides results that are either state of the art or improve upon it, for most quality measures.

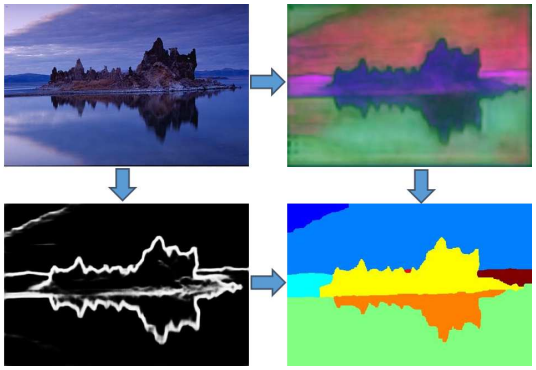


Figure 1: The algorithm flow: the proposed region representation (top right) and the edges (bottom left) of the input image (top left) are extracted and then combined to the resulting segmentation (bottom right).

1. Introduction

Generic segmentation is the well-studied task of partitioning an image into parts that correspond to objects for which no prior information is available. Deep learning approaches to this task thus far have been indirect and relied on edge detection. The COB algorithm [21], for example, uses a learned edge detector to suggest a high-quality contour map, and then creates a segmentation hierarchy from it using the oriented watershed transform [2].

In this work, we follow the approach used for semantic segmentation: learning pixel-wise representations that capture region and segment characteristics. However, we apply this approach for the first time to the generic (non-semantic) segmentation task. This paper focuses on creating such representations, along with combining them with edge-based information to a generic segmentation algorithm improving the current state of the art.

Deep learning has been successfully used in a supervised regime, where the network is learned end-to-end on supervised tasks (e.g., classification [14], object detection [29], semantic segmentation [6], or edge detection [36]). Generic segmentation, however, cannot be formulated as such. Unlike semantic segmentation, the properties of regions inside segments of new ("test") images are not well specified with respect to the regions or the objects in a training set, and therefore, the direct classification approach does not apply.

The common task of face verification ([34], [31], [26]) shares this difficulty. Even if we learn on thousands of labeled faces, there may be millions of unseen faces that must be handled. To succeed, we may learn a model, or representation, that captures properties capable of distinguishing between different faces, even those not encountered in training. Similarly, in generic segmentation, the images to be partitioned might contain objects not seen in the training set. The problem is further complicated by the fact that the annotated segments have unknown semantic meaning (i.e., we do not know what objects or parts are marked).

Thus, we learn pixel-wise representations that express segment relatedness. That is, the representations are grouped together in representation space for pixels of the same segment, and kept further apart for pixels from different segments. We use a supervised algorithm which follows the principles of the DeepFace algorithm [34] but addresses the differences between the tasks. We compare it to

*Both authors contributed equally

the more common approach for deep representation learning (triplet loss [15]), and test it quantitatively and visually.

Our Boundaries and Region Representation Fusion (BRRF) algorithm combines the learned representation with edges (from an off-the-shelf edge detector), as illustrated in fig. 1.

Our contributions in this paper are as follows:

1. We present (the first) pixel-wise representation for generic segmentation. This representation captures segmentation properties and performs better than previous methods on a pixel pair classification task.

2. We present a new segmentation algorithm that uses both the proposed pixel-wise representations and the traditionally used edge detection. This algorithm achieves excellent results, and for some quality measures, significantly improves the state of the art.

2. Related Work

2.1. Generic segmentation

Generic segmentation has seen a broad range of approaches and methods. Earlier methods (e.g., [8]) rely on clustering of local features. Modern methods often rely on graph representations, in which pixels or other image elements are represented by graph nodes, and weights on the edges may represent the similarity or dissimilarity between them. Then, segmentation is carried out by cutting the graph into dissimilar parts [5, 32]. The bottom-up watershed algorithm [25], for example, merges the two elements with the least dissimilar nodes at each iteration. See also [11].

The OWT-UCM algorithm [2] uses edge detection to get reliable dissimilarities, and an oriented watershed transform to transform the graph into a hierarchical region tree. Combining multiple scales further improves performance [27]. This approach, coupled with a CNN edge detector, achieves the current state-of-the-art [19].

An efficient and elegant way to represent the hierarchical region tree is through an Ultrametric Contour Map (UCM) [1], also known as a Saliency Map [25]. The UCM assigns a value to each contour in the image so that a contour that persists longer in the hierarchy gets a higher value. Thresholding the map provides a specific segmentation.

2.2. Representation learning

Representations can be learned explicitly using a Siamese network [7, 17, 13]. An example is a pair of inputs, either tagged as same (positive example) or not same (negative example). Both inputs are mapped to a representation through neural networks whose weights are tied. The networks are trained to minimize the distance in representation space between positive examples and increase the distance between negative examples. The learning can also be made through a triplet setting [15, 31], where an example is a triplet of inputs. The first two inputs are positive and negative, respectively, relative to the third one.

The representation can also be learned implicitly by learning a supervised task. The last layer of a network can be regarded as a classifier, while the rest of the network generates a representation that is fed to this classifier [10]. These representations can be used later on to distinguish between unseen classes [34], or for transfer learning [10].

The closest work to the representation part of this paper is Patch2Vec [12], which learns an embedding for image patches by training on triplets tagged according to the segmentation. Our approach differs by using implicit learning and allowing a larger context from the full image. Superpixel representations for generic segmentation are learned in [20]. This work differs from our per-pixel representation. As we will show later on, being pixel-based is important because it allows us to give consideration to edges, and construct the effective similarity that is later used. Besides, it is learned differently, using contrastive loss in a Siamese setting, and leads to segmentation results that are not as good as those achieved in recent works (including ours).

3. Representing Non-Semantic Segmentation

One well-known strength of neural networks is their ability to capture both low-level and high-level features of images, creating powerful representations [37, 4]. In this work, we focus on segmentation-related representations and harness this strength to provide a new pixel-wise representation. This learned representation should capture the segment properties of each pixel, so that representations associated with pixels that belong to the same segment are close in representation space, and their cluster is farther away from clusters representing different segments. This representation is a pixel-wise N-dimensional vector (thus, the full image representation, denoted R, is an $H \times W \times N$ tensor).

Learned classifiers have been used in the context of semantic (model-based) segmentation. Learning a classifier directly is possible for the semantic segmentation task because every pixel is associated with a clear label: either a specific category or background. This is not the case with generic segmentation, where object category labels are not available during learning and are not important at inference. Moreover, at inference, the categories associated with the segments are not necessarily those used in training.

3.1. Learning the representation

A pixel-wise representation for generic segmentation can be learned explicitly by minimizing a triplet loss over triplets [15, 31]. While this approach has proved beneficial for high-level image representations or patches, it has not been explored on tasks which provide structured outputs

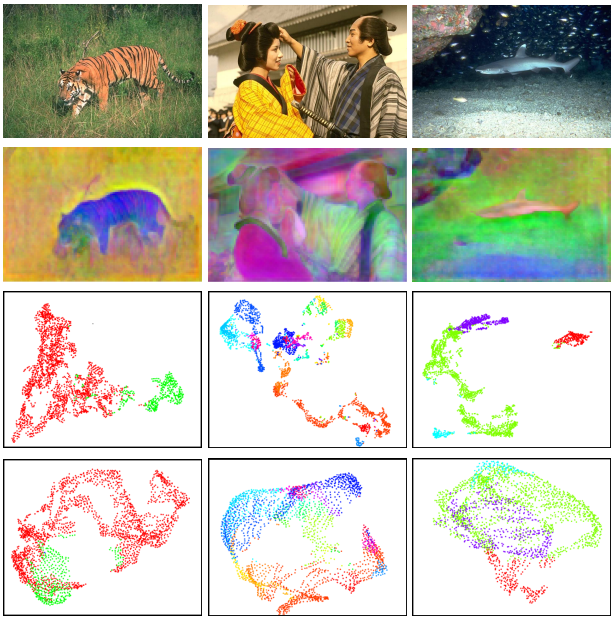


Figure 2: In each column, top to bottom: original image, representation space virtual colors, t-SNE of our representation, and t-SNE of the representation from a network trained for semantic segmentation. In t-SNE plots, points of the same color belong to the same segment. Notice how areas such as the tiger or the woman's kimono are nearly uniform in color, indicating that those pixels are close in representation space. Note that the t-SNE plots formed from our representation are better separated than the alternative.

such as segmentation. We shall compare to this approach later in this paper.

Representations can also be learned implicitly, by training a network on a related high-level task, and afterwards using the representation from the last hidden layer for the original task. However, using such a learning approach for generic segmentation related representations is not straightforward. To formalize such a high-level task, like classification, the labels must have some semantic meaning. For example, in the DeepFace approach [34] for face verification in the wild, an L-layer classification network is trained with K (> 4000) face identity classes, and the N-dimensional response from layer $L-1$ is used as the representation of the input image.

Training with cross-entropy inherently forms clusters of face images belonging to the same identity [10]. With sufficiently large number of face images, the network generalizes well and generates well-clustered representations even for new images of unseen categories. This usage is reminiscent of our generic segmentation task, where the aim is to separate pixels that belong to different segments not seen in training. We adopt this approach for learning a representation for generic segmentation. However, some differences need to be addressed. First, we need a representation for each image pixel rather than a single representation for a full image. We therefore use a fully convolutional network that outputs an N-dimensional vector for every output pixel.

A more fundamental difference is that choosing the training labels is not straightforward. Pixels in segmentation examples are assigned labels depending on the segment they belong to, but unlike face identities, the labels associated with different segments are not meaningful in the sense that they are not associated with object categories or even with appearance types. This creates an ambiguity in the given labeling. For example, in a certain image, label #1 refers to a horse, and in a different image, label #1 refers to the sky. To address this problem, we consider the set of segments from all images in the training set as different categories. That is, we assign a unique label l_k to all pixels in the i-th segment of the jth image (s_{ij}), a label that no other pixel in another segment or image is assigned. A visualization of the labeling process can be seen in Fig. 3.

However, the use of arbitrary categories leads to several difficulties. Two segments of different images may correspond to the same object category and may be very similar (e.g., two segments containing blue sky) but they are considered to be of different classes. Because the two segments have essentially the same characteristics, training a network to discriminate between them would lead to representations that rely on small differences in their properties or on arbitrary properties (e.g., location in the image), both leading to poor generalization. To overcome this difficulty, we modify the training process: when training on a particular image, we limit the possible predicted classes only to those that correspond to segments in this image, and not to the segments in the entire training set.

The fully convolutional NN is trained as a standard pixel-wise classification task which, when successful, will classify each pixel to the label of its segment. In that case, the network will have learned representations which are well-clustered for pixels in the same segment, and different for pixels in different segments. We denote the representation of the i-th pixel by R_i. The distance $||R_i - R_j||$ between

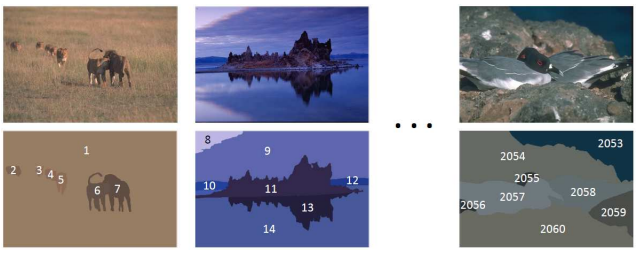


Figure 3: Our labeling process. We assign a unique label l_k to each segment in every training image.

two pixels should reflect the segment relatedness.

3.2. The representation learning network

The network architecture we use for the representation learning is a modified version of ResNet-50 [14]. We make use of layers $conv1$ through $conv5\_3$. To increase the spatial output resolution, we adopt two common approaches: first, atrous (or dilated) convolutions [6] are used throughout layers $conv5\_x$. Second, we use skip connections of layers $conv3\_4$, $conv4\_6$ and concatenate them with layer $conv5\_3$, upsampling all to the resolution of $conv3\_4$, which is 4 times smaller than the original input resolution. The concatenated layers pass through a final $fuse$ residual layer, to get a final feature depth of 512 per pixel. Fig. 4 illustrates the network architecture (referred to as RepNet).

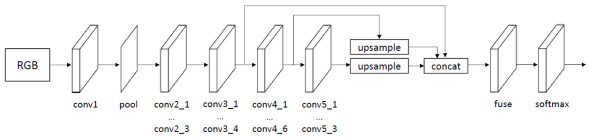


Figure 4: Our suggested RepNet architecture.

In general, the huge pixel-wise classification task (thousands of classes for each pixel) is a major hardware bottleneck that limits our resolution upsampling capabilities. While semantic segmentation task architectures are able to upsample to the original input resolution (the number of classes are in the range of tens), we were limited to a resolution of $\frac{1}{4}\times$ of the original input image. The upsampling is done with bilinear interpolation. We found that here, deconvolution-based upsampling did not improve performance.

The network was trained using a weighted cross-entropy loss. To improve segment separation, we increased the weight of the loss associated with pixels close to the boundary (closer than d) by a factor of w_b. The proposed pixel-wise representation is taken from the final layer before softmax, which we refer to as $fuse$.

3.3. Visualizing the representations

We suggest two visualization options:

1. **Representation space virtual colors –** We project the N-dimensional representations on their three principal components (calculated using the PCA of all representations). The three-dimensional vector of projections is visualized as an RGB image. We expect pixels in the same segment to have similar projections and similar color. Fig. 2 shows this is indeed the case.

2. **t-SNE scatter diagram –** Intuition about the representations can also be gained by using t-Distributed Stochastic Neighbor Embedding (t-SNE, [35]) to embed them in a 2D space. We expect a separation between points belonging to different segments in the embedded space, and compare the separation to that obtained by a network with the same architecture but trained for semantic segmentation. See Fig. 2.

Note that the representation changes sharply along boundaries (Fig. 2). This behavior is typical to nonlinear filters (e.g., bilateral filter) and therefore indicates that the representation at a pixel depends mostly on image values associated with the segment containing the pixel and not on image values at nearby pixels outside this segment. That is, the representation describes the segment and is not a simple texture description associated with a uniform neighborhood.

4. A Hierarchical Segmentation Algorithm

Many modern and successful hierarchical segmentation algorithms work by agglomerating image elements, based on high-quality edge detection results [19, 21, 2].

To demonstrate the utility of the proposed representation we incorporate it in the agglomerative approach. We use the representation twice. First, for recalculating the dissimilarity value for a pair of segments, and then for re-ranking these values using region context. Our suggested Boundaries and Region Representation Fusion (BRRF) algorithm for hierarchical image segmentation consists of three stages:

1. Initialization: Creating an initial oversegmentation and calculating the proposed region representations.

2. Using a classifier to calculate an augmented pair dissimilarity for every two neighboring segments.

3. Iteratively, merging pairs of segments and recalculating dissimilarities using region context.

4.1. Algorithm initialization

We obtain an initial oversegmentation using the oriented edge detection results and the watershed algorithm [2, 21]. We remove small superpixels (SPs) by iteratively combining every SP smaller than 32 pixels with its most similar neighbor - the one with the lowest average edge strength in their shared boundary. This is a common practice in agglomerative segmentation and is necessary here due to the representations being inaccurate for very small SPs.

Independently, we calculate the representation as described in Section 3.2. To improve performance on small segments, we upsample the representation, through bilinear interpolation, to half of the original image resolution. For computational reasons, we reduce the representation's dimension to 9 using PCA (calculated separately for this image). This transformation is effective because representations in a specific image lie in a low dimension subspace.

4.2. Pair dissimilarity calculation

To incorporate the representation into the agglomerative process, we use a learned dissimilarity function which depends on the representation, edge strength, and some additional properties that describe the segments' geometry and raw color. This dissimilarity, denoted $Pair\_Dissim$, is calculated for every pair of neighbouring segments. A higher dissimilarity indicates that the two segments are more likely to belong to different (semantic) objects. We elaborate on learning the dissimilarity in Section 5.

4.3. The iterative merging stage

This stage is iterative. At every iteration, we merge two segments into one and then recalculate the dissimilarity between this merged segment and its neighbors.

The merged segments may be chosen as the pair associated with the lowest $Pair\_Dissim$ in the current hierarchy. This works well (see Section 6.3), but an improved decision is achieved by relying also on a context-based clustering test, which uses the representation as well.

The Silhouette score [30] is an unsupervised measure of cluster separation. It calculates the dissimilarities between elements contained in one cluster and other elements in other clusters, and compares these dissimilarities to the dissimilarity within the cluster. It achieves a high score when the former is high and the latter is low, indicating that the clusters are well separated. Inspired by the Silhouette score, we consider a segment pair and test whether the union of the two segments is a well separated cluster.

Denote the candidate merged segment by S, and the union of all its neighboring segments (in the current segmentation), by S_n; see an example in Fig. 5. Let $d(R_i, R_j)$ be the Euclidean distance between the representations R_i, R_j, corresponding to pixels i, j respectively. Then, the dissimilarities $a(i), b(i)$ associated with pixel $i \in S$, and the corresponding Silhouette score are:

$$
\begin{aligned}
a(i) &= \frac{1}{|S|-1} \sum_{j \in S, j \neq i} d(R_i, R_j) \\
b(i) &= \frac{1}{|S_n|} \sum_{j \in S_n} d(R_i, R_j) \\
Sil\_Score &= \frac{1}{|S|} \sum_{i \in S} \frac{b(i) - a(i)}{max\{a(i), b(i)\}}
\end{aligned}
\tag{1}
$$

For large segments, only a subset of the segment pixels are used for both S and S_n; see Section 5.3.

This criterion is not accurate enough to be used by itself, as an alternative to the *Pair_Dissim*. This holds especially in the beginning of the merging process, when the segments are small, and are typically not very dissimilar to their neighbors. Therefore, we do not calculate the silhouette score until we have less than $\#s = 120$ segments. Then, at each iteration, we consider only the pairs that achieve the minimal $T(= 4)$ pair dissimilarity and merge the pair which achieves the minimal augmented dissimilarity,

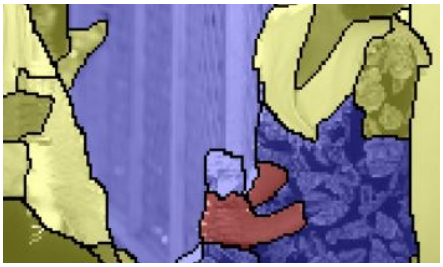


Figure 5: The Silhouette score calculation. The merged segment candidate S (red), the union of its neighboring segments S_n (blue), and process irrelevant segments (yellow).

$$
Aug\_Dissim = Pair\_Dissim - 0.5 \cdot Sil\_Score. \tag{2}
$$

(Note that high *Sil_Score* is preferred for merging.) We refer to this process as re-ranking.

4.4. Parametrizing the hierarchy

The iterative merging process creates a segmentation hierarchy, parameterized by UCM values that should be monotonic with the iteration number (Section 2.1, [1]). This is the case with edge confidence [2, 21], but not necessarily with our dissimilarity values; see Fig. 1 in Appendix A.

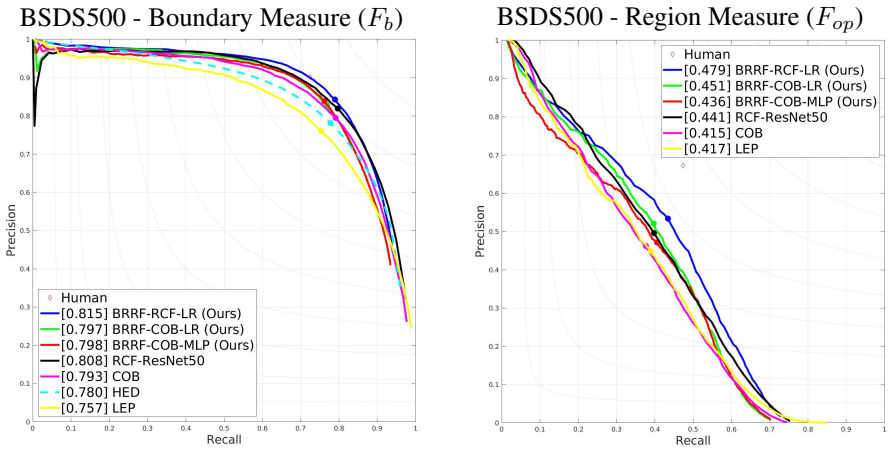
Using the iteration number itself [9] is problematic: It is not uniformly correlated with segmentation quality over different images. Therefore, controlling the segmentation coarseness by uniform, image-independent thresholding, gives uneven segmentations (and low ODS F-score). We used a monotonized value of the *Pair_Dissim*, and got excellent results. See Appendix A.

5. The Pair Dissimilarity Classifier

5.1. Pair Dissimilarity features

We use three types of features for our classification task:

Representation-based features - The first type is 10 features calculated from the representations of the two associated segments. We use only a subset of pixel representations from the segments (see Section 5.3). For every pair of pixels, one from each segment, we calculate the minimum, maximum, average, and median of both the L2 and the cosine distances between them (following [16]), as well as distances between the calculated mean of the representation in each segment. The L2 distances are replaced by their log values, to bring all distance calculated to the same range. This choice of features follows classic agglomerative clustering methods [33].



Method	ODS	F_b OIS	AP
BRRF-RCF-LR (Ours)	**0.815**	**0.838**	0.855
BRRF-COB-LR (Ours)	0.797	0.820	0.835
BRRF-COB-MLP (Ours)	0.798	0.819	0.832
RCF-ResNet50 [19]	0.808	0.833	**0.873**
COB [21]	0.793	0.820	0.859
HED [36]	0.780	0.796	0.834
LEP [38]	0.757	0.793	0.828

Method	ODS	F_{op} OIS	AP
BRRF-RCF-LR (Ours)	**0.479**	**0.531**	**0.390**
BRRF-COB-LR (Ours)	0.451	0.513	0.359
BRRF-COB-MLP (Ours)	0.436	0.503	0.338
RCF-ResNet50 [19]	0.441	0.5	0.370
COB [21]	0.415	0.466	0.333
LEP [38]	0.417	0.468	0.334

Figure 6: BSDS Test evaluation: Precision-recall curves for evaluation of boundaries [23], and regions [28]. Open contour methods in dashed lines and closed boundaries (from segmentation) in solid lines. ODS, OIS, and AP summary measures. Markers indicate the optimal operating point, where Fb and Fop are maximized.

Edge-based feature - The averaged edge probability scores between the segments from an edge detection algorithm [21, 19].

Geometric and raw color features - The first 3 features describe the geometry of the merged segments: the (square root of the) length of the boundary between the segments, the (square root of the) combined segment size, and the maximum ratio of the boundary (between the segments) length and the segment's perimeter.

The last 3 features describe the color dissimilarity and are specified as the 3 differences between the averages of the LAB color space values in the two segments.

These 17 features serve as input to the classifier that outputs the $Pair\_Dissim$. While edge detection is very informative, our representations encode information about the properties associated with high-level segment content, hence it further boosts performance.

5.2. Training the classifier

To train the classifier, we use segment pairs taken from the segmentation hierarchies generated by the used edge detection algorithm. From each hierarchy, we first generated several segmentations by thresholding the UCM with different values. We use all the neighboring segment pairs in a segmentation as either positive (i.e., should merge) or negative (should not merge) examples. A pair is considered negative when at least 0.6 of its shared boundary is close (within 2 pixels) to the boundary specified in at least one ground truth annotation. The threshold separating between positive and negative examples was empirically determined.

We experimented with several classifiers, and found that the best results were obtained with LR (Logistic Regression) and MLP (Multilayer Perceptron); see Appendix C.

5.3. Segment filtering and sampling

The representation is not completely uniform over each segment, and, in particular, includes outliers. Therefore, before using the representation vectors, we filter them using Isolation Forest [18]. For computational efficiency, we also sample large segments (> 300 pixels). Both filtering and sampling improve the accuracy; see details in Appendix B.

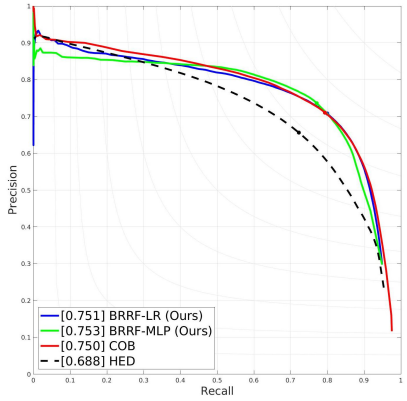
6. Experiments

We first present details of the representation learning procedure and the evaluation of these representations in a pixel pair classification task. We then briefly compare some versions of the agglomerative segmentation algorithm. Finally, we present quantitative and qualitative segmentation results. The BRRF algorithm runtime is 45 seconds per image (average). No attempt to optimize the code was done. Our code for both the RepNet and the BRRF algorithm is available on https://github.com/oranshayer/BRRF.

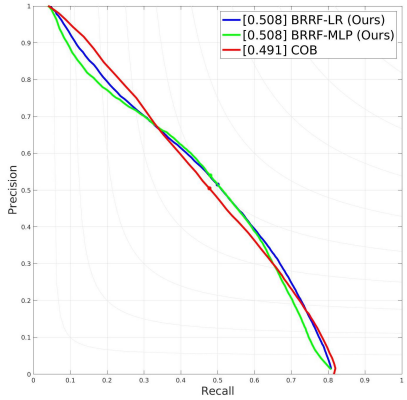
6.1. RepNet training details

We trained the representation network over the classification task described in Section 3.1 both for the Pascal Context dataset [24] and BSDS500 dataset [22]. For Pascal Context, we started with ImageNet pre-trained weights, and trained with 1000 images containing 5540 segments. We trained the network for 300 epochs with $d = 8$ and $w_b = 5$, SGD optimizer with momentum set to 0.9, and a learning rate of 0.01 which was divided by 10 after 250 epochs. For the BSDS500 dataset, the network was fine-tuned from the network trained on Pascal, with 300 *trainval* images containing 2060 segments. We started with a learning rate of 0.001 and trained for 300 epochs.

Pascal Context - Boundary Measure (F_b)



Pascal Context - Region Measure (F_{op})



Method	ODS	F_b OIS	AP
BRRF-COB-LR (Ours)	0.751	**0.785**	0.756
BRRF-COB-MLP (Ours)	**0.753**	0.781	0.750
COB [21]	0.750	0.781	**0.773**
HED [36]	0.688	0.707	0.704

Method	ODS	F_{op} OIS	AP
BRRF-COB-LR (Ours)	**0.508**	**0.580**	**0.439**
BRRF-COB-MLP (Ours)	**0.508**	0.579	0.427
COB [21]	0.491	0.565	**0.439**

Figure 7: Pascal Context Test evaluation: Precision-recall curves for evaluation of boundaries [23], and regions [28].

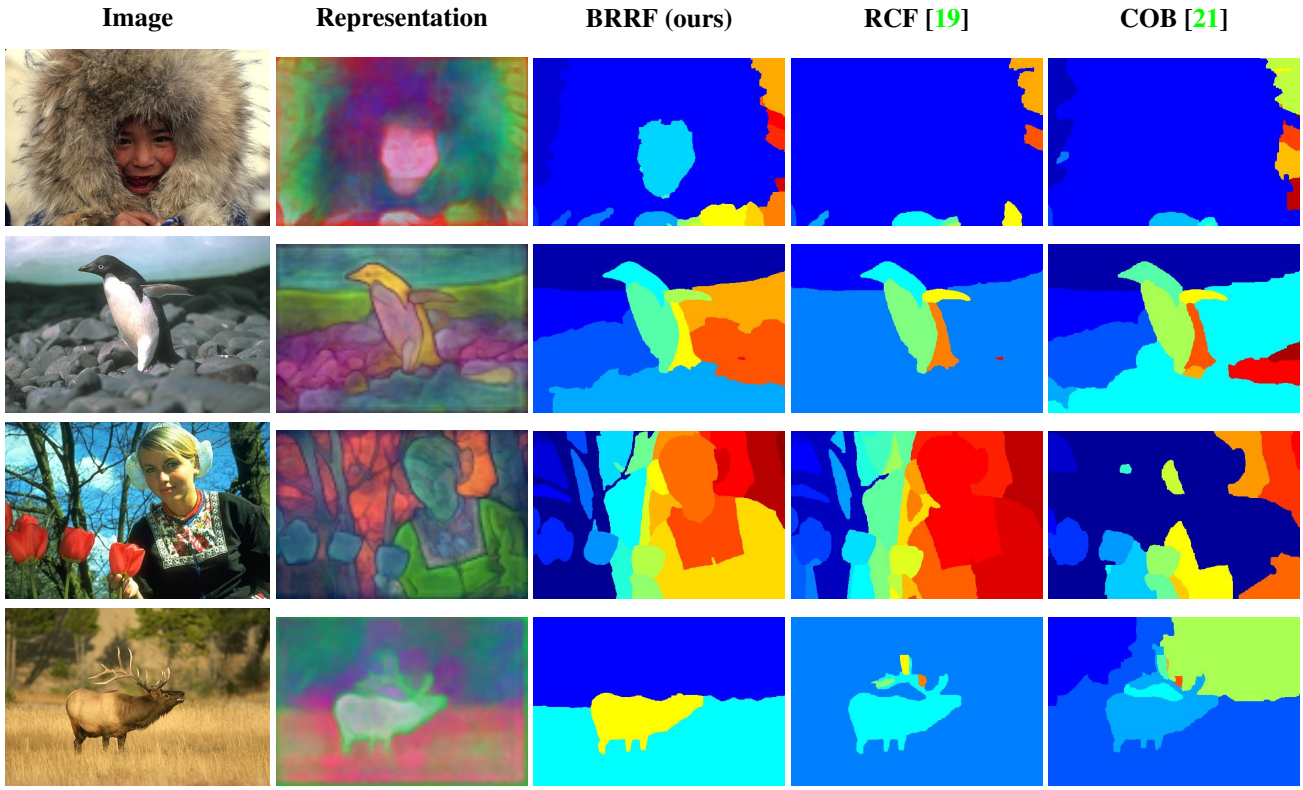


Figure 8: Some segmentation examples produced by our algorithms and other competitive algorithms. The segmentations were chosen by maximizing the F_{op} score in the hierarchy. Our algorithm achieves a clearer segmentation between different objects (e.g., the faces in rows 1 and 3); captures differences between segments of the same class (background in row 2); but struggles with narrow segments (the horns in row 4).

6.2. Evaluating the representations

To evaluate the quality of the representations themselves, we consider a pixel classification task. The task is to determine whether two pixels belong to the same segment using the representations. To this end, we use a simple classifier which decides that the pixels belong to the same segment if the Euclidean distance between the representations is smaller than a threshold. The optimal threshold is learned over a validation set. The results are presented in Table 1.

We compared the representations obtained from our implicit learning method with representations learned as follows: triplet loss, representation from a network trained for

semantic segmentation [6], and representations from a network trained for material classification where materials hold strong texture characteristics [3]. We also compare with the following pixel representations: RGB, L*a*b, and Gabor filters. Clearly, our proposed representation achieves the best result, suggesting it can be beneficial for pixel separation tasks such as image segmentation.

Representation	Test accuracy
Gabor filters	56.09%
RGB	57.67%
L*a*b	58.25%
Material classification net [3]	70.14%
DeepLab [6]	71.94%
Triplet loss	76.34%
Ours (implicit learning)	**81.04%**

Table 1: Pixel pair classification results

6.3. Comparing different merging functions

We experimented with several versions of the merging function and found it fairly robust to its parameters. See the more detailed description in Appendix C.

Different components of the pair dissimilarity - Table 2 shows the OIS measure of the F_{op} score. For each combination, the classifier was retrained, and then tested on the BSDS500 validation set. Adding the representation noticeably improves our score.

Re-ranking - Re-ranking required very little additional runtime, but improved accuracy (see Table 2 and Appendix C). Increasing $(\#s, T)$ beyond $(120, 4)$ did not result in further improvement.

Filtering and sampling - Filtering with the isolation filter improves accuracy (OIS F_{op} increases by 0.004) and increases runtime by 20%. Using less samples significantly speeds the algorithm and, somewhat surprisingly, slightly increases the accuracy.

6.4. Generic image segmentation

We evaluated our segmentation algorithm on both the BSDS500 dataset [22] and the Pascal Context dataset [24] (independently) for both LR and MLP classifiers using [21] for edge detection (denoted as BRRF-COB-LR and BRRF-COB-MLP, respectively). Additionally, we evaluated our algorithm on BSDS500 using [19] on a ResNet50 architecture for edge detection with an LR classifier (denoted as BRRF-CRF-LR). We trained both the network and merging classifier on the trainval sets on both datasets.

Edge	Geo. and color	Rep.	Re-ranking	F_{op}
x				0.453
x	x			0.484
x		x		0.49
x	x	x		0.509
x	x	x	x	0.512

Table 2: Features effect on the segmentation results

BSDS500: Results are presented in Fig. 6. Using an LR classifier and the [19] edge detection, we have either matched or improved the state-of-the-art results for the F_b score [23]. Noticeably, we have achieved recall and precision scores that match those of a human annotator. For the F_{op} score [28], we significantly improved the state of the art for both versions of the F-score and improved the average precision.

Pascal Context: The results are similar; see Fig. 7. Using an LR classifier, we have either matched or improved the state-of-the-art results on both versions (OIS and ODS) of F_b and F_{op}, with more significant improvement of the latter.

For both datasets, the proposed algorithm achieves a more substantial improvement on the region evaluation measures (F_{op}). This is expected because our algorithm differs by using region representations. While we get the best results in nearly all the F_b, F_{op} scores, this is not always the case with the AP score, which is influenced also by extreme oversegmentation and undersegmentation

7. Conclusion

We proposed a new approach to generic image segmentation that combines the strengths of edge detection and (a new) pixel-wise representation. The representation, learned through a formulation of a suitable supervised learning task, is a region representation. As such, it complements edge information and improves the segmentation quality, especially when it comes to region-based quality measures.

Experimentally, the proposed representation, by itself, achieves excellent, state-of-the-art results in pixel pair classification, and overcomes earlier learned and non-learned representations, serving as evidence that it captures characteristics that distinguish between different segments and generalizes well for segments not seen in the training set. The use of these representations through a complete segmentation algorithm yields state-of-the-art results for generic segmentation.

References

[1] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 182–182. IEEE, 2006.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[3] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 539–546, 2005.

[8] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. volume 24, pages 603–619, 2002.

[9] J. Cousty, L. Najman, Y. Kenmochi, and S. Guimarães. Hierarchical segmentations with graphs: quasi-flat zones, minimum spanning trees, and saliency maps. *Journal of Mathematical Imaging and Vision*, 60(4):479–502, 2018.

[10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[11] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, Sept. 2004.

[12] O. Fried, S. Avidan, and D. Cohen-Or. Patch2vec: Globally consistent image patch representation. In *Computer Graphics Forum*, volume 36, pages 183–194. Wiley Online Library, 2017.

[13] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[15] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[16] S. Horiguchi, D. Ikami, and K. Aizawa. Significance of softmax-based features in comparison to distance metric learning-based features. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[17] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[18] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[19] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang. Richer convolutional features for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(8):1939–1946, 2019.

[20] Y. Liu, P.-T. Jiang, V. Petrosyan, S.-J. Li, J. Bian, L. Zhang, and M.-M. Cheng. Del: Deep embedding learning for efficient image segmentation. In *IJCAI*, 2018.

[21] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[22] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[23] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, May 2004.

[24] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[25] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 18(12):1163–1173, 1996.

[26] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[27] J. Pont-Tuset, P. Arbeláez, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. In *arXiv:1503.00848*, March 2015.

[28] J. Pont-Tuset and F. Marques. Supervised evaluation of image segmentation and object proposal techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(7):1465–1478, 2016.

[29] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[30] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[31] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society, 2015.

[32] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, Aug. 2000.

[33] P. H. Sneath, R. R. Sokal, et al. *Numerical taxonomy. The principles and practice of numerical classification.* 1973.

[34] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1701–1708, Washington, DC, USA, 2014. IEEE Computer Society.

[35] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9: 25792605, Nov 2008.

[36] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015.

[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[38] Q. Zhao. Segmenting natural images with the least effort as humans. In M. W. J. Xianghua Xie and G. K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 110.1–110.12. BMVA Press, September 2015.