DEL: Deep Embedding Learning for Efficient Image Segmentation

Yun Liu[1], **Peng-Tao Jiang**[1], **Vahan Petrosyan**[2],
Shi-Jie Li[1], **Jiawang Bian**[3], **Le Zhang**[4], **Ming-Ming Cheng**[1*]
[1] Nankai University
[2] KTH Royal Institute of Technology
[3] University of Adelaide
[4] Advanced Digital Sciences Center
nk12csly@mail.nankai.edu.cn, cmm@nankai.edu.cn

Abstract

Image segmentation has been explored for many years and still remains a crucial vision problem. Some efficient or accurate segmentation algorithms have been widely used in many vision applications. However, it is difficult to design a both efficient and accurate image segmenter. In this paper, we propose a novel method called DEL (deep embedding learning) which can efficiently transform superpixels into image segmentation. Starting with the SLIC superpixels, we train a fully convolutional network to learn the feature embedding space for each superpixel. The learned feature embedding corresponds to a similarity measure that measures the similarity between two adjacent superpixels. With the deep similarities, we can directly merge the superpixels into large segments. The evaluation results on BSDS500 and PASCAL Context demonstrate that our approach achieves a good trade-off between efficiency and effectiveness. Specifically, our DEL algorithm can achieve comparable segments when compared with MCG but is much faster than it, *i.e.* 11.4fps *vs.* 0.07fps.

1 Introduction

Image segmentation aims to partition an image into large perceptual regions, where pixels within each region usually belong to the same visual object, object part or large background region with tiny feature difference, *e.g.* color, gradient, texture, and intensity. Image segmentation has been widely used in mid-level and high-level vision tasks, such as object proposal generation [1; 2], tracking [3], object detection/recognition [4], semantic segmentation [5], and so on. This technique has been studied for many years, but still remains a main challenge in computer vision.

In general, image segmentation addresses two aspects, the reliability of segmentation results and efficiency for applications. An appropriate segmented image can be used as input to significantly improve the performance of many vision tasks. Furthermore, the computational time and memory consumption determine whether it is suitable for many practical applications or not, because image segmentation is often used as a preprocessing step in other vision applications. However, it is difficult for existing methods to balance the segmentation accuracy and computational time. Although MCG [1] and gPb [6] can generate high-quality segments, they are too slow to be applied in time-sensitive tasks. The running time of EGB [7] is nearly proportional to the number of image pixels, so it is very fast. But it suffers poor accuracy especially on the region evaluation metric and thus can not satisfy today's vision tasks. HFS [8] can perform real-time segmentation. However, the segmentation results are not satisfactory, especially on the region evaluation metric. It is difficult to design an ideal algorithm that can simultaneously satisfy the requirements of effectiveness and efficiency.

Similar but slightly different from image segmentation, superpixel generation usually refers to an oversegmentation. It segments an input image into small, regular and compact regions, which is distinct from the large perceptual regions generated by image segmentation techniques. Oversegmentation usually has strong boundary coherence, and the number of produced superpixels can be easy to control. Since superpixel methods are usually designed to generate small segments, it is inappropriate to directly use them to generate large regions. However, superpixel algorithms provide a good start for image segmentation.

In this paper, we aim to design an image segmentation algorithm that can make a good trade-off between efficiency and effectiveness. Considering the efficiency, our effort starts with a fast superpixel generation method, the GPU version of SLIC [9; 10]. In the past few years, convolutional neural networks have pushed the boundaries of many computer vision tasks. Since deep features can represent much richer information than hand-crafted features, we train a fully convolutional network to learn the deep feature embedding space that encodes the deep representation of each superpixel. We introduce a deep embedding metric that converts the feature embedding vectors of adjacent superpixels to a similarity value. Each similarity value represents the probability that two adjacent superpixels belong to the same region. By this way, we can train the deep embedding space end-to-end to learn

*M.M. Cheng is the corresponding author.

the similarity between each pair of superpixels. A novel network that combines the features of fine details from bottom sides and high-level information from top sides is proposed for the embedding learning. We merge the adjacent superpixels into large image segments if the learned similarities between them are larger than a threshold. This simple merging operation can achieve better performance than HFS's hierarchical merging due to the powerful representation of deep features.

We conduct extensive experiments on BSDS500 [6] and PASCAL Context [11] datasets to evaluate the proposed image segmentation algorithm. To evaluate our algorithm in applications, we apply our segmentation results to object proposal generation on the PASCAL VOC2007 dataset [12]. The evaluation results demonstrate that our algorithm achieves a good trade-off between efficiency and effectiveness. Specifically, our proposed DEL can achieve comparable segmentation results when compared with state-of-the-art methods, while much faster than them, *e.g.* 11.4fps of DEL *vs.* 0.07fps of MCG. This means DEL has the potential to be used in many practical applications. The code of this paper is available at https://github.com/yun-liu/del.

1.1 Related Work

In the past several decades, researchers have contributed lots of useful works to this field. Due to the limitation of space, we only review some typical algorithms here. Shi *et al.* [13] viewed image segmentation as a graph partitioning problem. A novel Normalized Cut criterion was proposed to measure both the total similarity within each segment and the total dissimilarity between different segments. Comaniciu *et al.* [14] proposed Mean Shift, based on the old pattern recognition procedure of mean shift. Felzenszwalb *et al.* [7] proposed an efficient graph based algorithm, EGB. The edge-based method, gPb [6], combines multiscale local features and spectral clustering to predict edges and then converts these edges to a segmentation using an oriented watershed transform algorithm. Pont-Tuset *et al.* [1] combined multiscale hierarchical regions to acquire accurate segmentation (MCG).

With the development of superpixel generation [9], some methods attempt to start with superpixels and then merge these superpixels into perceptual regions. ISCRA [15] uses gPb to generate high-quality superpixels. A dissimilarity score is learned for adjacent superpixels using various features, including color, texture, geometric context, SIFT, shape, and boundary. Cheng *et al.* [8] firstly built a real-time image segmentation system by a hierarchical merging of superpixels using carefully selected parallelizable features. The combination weights of selected features are retrained at each merging stage. Our proposed method falls into this category, too. But our method uses a deep convolutional neural network to extract powerful deep representation for this task, and thus can obtain better perceptual regions. We will introduce our method in detail in the next section.

2 Our Approach

Our approach starts with SLIC [9] superpixels. We first train a deep network to learn similarities between neighboring superpixels, and then directly merge them using the learned similarities. In this section, we will describe our algorithm's five components in detail, which are superpixel generation, feature embedding learning, network architecture, superpixel merging, and implementation details in order.

2.1 Superpixel Generation

Image segmentation algorithms group pixels into large perceptual regions, where pixels in the same region have greater similarities than pixels in different regions. However, when grouping pixels using similarity distance metrics, the algorithms usually consume too much time because the running time of the algorithm is highly related to the number of pixels in an image. Furthermore, the algorithm lacks robustness when directly merging pixels. Considering these two aspects, our algorithm starts with a fast superpixel generation method, SLIC [9], which is based on the k-means clustering algorithm. The number of superpixels is much less than the original pixels, so this makes it possible to improve efficiency. One superpixel is a small region, and thus more robust than single pixels.

In general, superpixel algorithms cannot be directly applied to image segmentation because large perceptual regions are usually not regular and related to the global information in an image, unlike superpixels. Inspired by HFS that starts with superpixels and uses carefully designed features to merge them hierarchically, our algorithm learns a similarity metric between adjacent superpixels. SLIC is one widely used superpixel algorithm among many state-of-the-art algorithms due to its simplicity and efficiency. We choose the GPU version of SLIC, gSLIC[10], as the start of our method. In order to balance the running time and the boundary adherence of the generated superpixels, we control each superpixel to contain about 64 pixels. Suppose we have M superpixels for an image I now. The set of generated superpixels is denoted as $\mathcal{S} = \{S_1, S_2, \cdots, S_M\}, S_i = \{1, 2, \cdots, |I|\}^{|S_i|}$.

2.2 Feature Embedding Learning

After generating superpixels, we start to train a deep convolutional neural network to learn the feature embedding space. As shown in Figure 1, we perform pooling operation on the feature embedding space to get feature vectors $\vec{v} = \{\vec{v}_1, \vec{v}_2, \vec{v}_3, \cdots, \vec{v}_M\}$ corresponding to the superpixels. Each feature vector is the average of the learned deep feature maps in the corresponding region of the superpixel. It can be formulated as follows:

$$\vec{v}_i = \frac{1}{|S_i|} \sum_{k \in S_i} \vec{x}_k, \tag{1}$$

where $\vec{x}_k$ denotes the feature vector within the region of the i-th superpixel. We call this pooling operation superpixel pooling. Each feature embedding vector $\vec{v}_i$ has 64 dimensions in our design. The backwards function of the superpixel pooling layer with respect to input $\vec{x}_k$ can be written as

$$\frac{\partial L}{\partial \vec{x}_k} = \sum_{S_i \in \mathcal{S}} 1_{\{k \in S_i\}} \cdot \frac{1}{|S_i|} \cdot \frac{\partial L}{\partial \vec{v}_i}, \tag{2}$$

in which $1_{\{k \in S_i\}}$ is an indicator function.

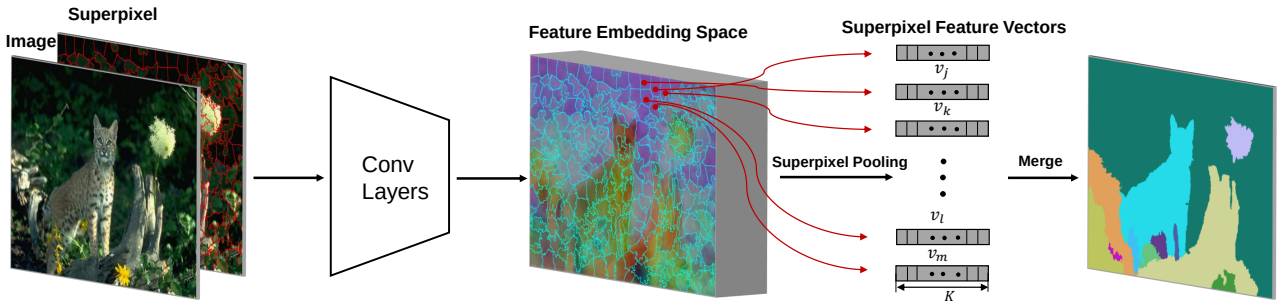

Figure 1: The pipeline of our DEL image segmentation algorithm.

We design a distance metric to measure the similarities between adjacent superpixels. The proposed distance metric can be formulated as

$$d_{ij} = \frac{2}{1 + exp(\|\vec{v}_i - \vec{v}_j\|_1)}. \tag{3}$$

The similarity $d_{i,j}$ ranges in [0, 1]. It is close to 1 when v_i and v_j are similar, and is close to 0 when v_i and v_j are extremely different. Since the distance metric is established, we consider the similarity loss function as follows:

$$\begin{aligned} L = -\sum_{S_i \in \mathcal{S}} \sum_{S_j \in \mathcal{R}} [(1-\alpha) \cdot l_{ij} \cdot log(d_{ij}) \\ + \alpha \cdot (1 - l_{ij}) \cdot log(1 - d_{ij})], \end{aligned} \tag{4}$$

where $l_{ij} = 1$ denotes v_i and v_j belongs to the same region, and $l_{ij} = 0$ denotes v_i and v_j belongs to different regions. $\mathcal{R}$ is the set of adjacent superpixels of the superpixel S_i. $\alpha = |Y_+|/|Y|$, denotes the proportion of the pairs of superpixels belonging to the same regions in the ground truth. We use this parameter to balance the positive samples and negative samples.

Using this similarity loss, we can learn the feature embedding space in an end-to-end manner. The similarities between the pairs of superpixels in the same ground truth segments are expected to be larger than the similarities of the pairs of superpixels belonging to different segments. In the next step, we will use the learned similarity distance metric to merge these superpixels.

2.3 Network Architecture

In this section, we introduce our network architecture which is used to learn the feature embedding space. Our network is built based on the VGG16 net [16] and inspired by recent works [17; 18]. The convolutional layers in VGG16 can be divided into five convolution stages by the pooling layers. As shown in Figure 2, we cut the *pool5* layer and the fully connected layers of the VGG16 Net. Because of the low resolution of the side output from the *conv5* stage, we modify the stride of *pool4* from 2 to 1. The hole algorithm [19] is used to keep receptive field sizes of convolutional layers in the fifth stage the same as the original VGG16 network. We consider that the learned features become coarser and coarser when the network goes deep. The fine features contain more detail information meanwhile the coarse features represent global information. The features from five stages are concatenated

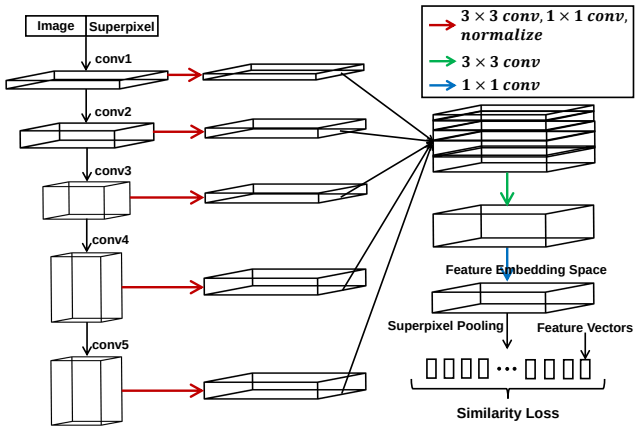

Figure 2: The network architecture for feature embedding learning.

to combine the coarse and global information with the fine and local information.

Specifically, we connect a 3×3 convolution layer on side-1-5 with 32, 64, 128, 256, 256 output channels, respectively. After the 3×3 convolution layer, a 1×1 convolution layer is connected with 32, 64, 64, 128, 128 output channels for side-1-5, respectively. Since the feature scales of different convolution stages are different, direct concatenating features from multiple stages will make features from some stages no sense. Therefore, we normalize the responses of different stages using the L2 normalization technique introduced in [20]. After the normalization, we concatenate the feature maps from all sides and a 3×3 convolution layer with 256 output channels is followed. Finally, we get the 64-dimension feature embedding space using a convolutional layer with kernel size 1×1. As illustrated in section3.2, we pool the feature embeddings to feature vectors corresponding to the superpixels, then use the proposed similarity loss to train the network.

2.4 Superpixel Merging

The dissimilarities between adjacent superpixels learned by the deep neural network are used to merge the superpixels into perceptual regions. A threshold is set to determine whether two adjacent superpixels should be merged or not. The pseudo code of superpixel merging is displayed in Algorithm 1. For the efficiency of merging, we use the data structure of *universe* in EGB [7]. Unlike the hierarchical merging strategy in HFS [8], we perform merging operation only once. HFS uses a linear combination of some low-level features and

retrains the combinatorial weights at each merging stage. The single-stage merging of DEL can also outperform HFS by a large margin. We will show the details in the experiment part.

Algorithm 1 Superpixel merging algorithm of DEL

Input: Image I, dissimilarity $f = (1 - d)$, threshold T, superpixels $\mathcal{S} = \{S_1, S_2, \cdots, S_M\}$
Construct $\mathcal{R} = \{R_1, R2, \cdots, R_M\}$, in which R_i is the set of adjacent superpixels of S_i
for each $S_i \in \mathcal{S}$ **do**
  for each $S_j \in R_i$ **do**
    if $f_{i,j} < T$: **then**
      $S_i \leftarrow S_i \cup S_j, \mathcal{S} \leftarrow \mathcal{S} \backslash S_j$
      Update $\mathcal{R}$
    end if
  end for
end for
Output: Segmentation $\mathcal{S}$

2.5 Implementation Details

Our network is based on Caffe, which is a widely used deep learning framework. Generally speaking, the segmented regions usually refer to the visual objects, object parts or the partial background. Therefore, we firstly pretrain our network for the semantic segmentation task on the SBD [21] dataset to acquire semantic information for the network. The network is tuned by replacing the feature embedding space with a classification layer for semantic segmentation task.

We then fine-tune the pretrained model for the feature embedding space. We use the stochastic gradient descent (SGD) technique to optimize the neural network. The basic learning rate is set to 1e-5. We use a weight decay of 0.0002 and batch size of 5. The learning rate policy of *step* is used, and we totally run SGD for 10000 iterations with *step size* of 8000. The learning rate of the feature embedding layer is set larger than the basic convolutional layers as suggested in deep metric learning.

Data augmentation has been proven to be important for deep learning. When training our feature embedding model on the BSDS500 dataset [6] that consists of 300 *trainval* images and 200 *test* images, we augment the *trainval* set. The images are rotated to 16 direction angles and also flipped at each angle. We then crop the largest rectangle from the transformed images, resulting in 9600 training images. When training on the PASCAL Context dataset that is divided into 7605 *trainval* images and 2498 *test* images, we only flip the *trainval* images for training because the number of images in this set is adequate.

3 Experiments

In this section, we first evaluate our DEL method on the BSDS500 dataset [6] and the PASCAL Context dataset [11] for image segmentation. In order to evaluate the segmentation quality in applications, we use the segmented regions to generate object proposals on the PASCAL VOC2007 dataset [12]. For the evaluation of image segmentation, we use the publicly available benchmark SEISM [22]. Optimal dataset scale (ODS) usually refers to the best performance when selecting optimal parameters for the whole dataset, while optimal image scale (OIS) refers to the best performance when selecting special parameters for each image. We report the boundary F-measure (F_b) and region F-measure (F_{op}) at ODS and OIS. For the evaluation of object proposals, we report the detection recall (DR) when varying the number of proposals. We compare our DEL with some state-of-the-art segmentation algorithms, including EGB [7], Mean Shift [14], NCuts [23], gPb-UCM [6], MCG [1], SLIC [9], GPU-SLIC [10], and HFS [8]. In addition to the GPU version of SLIC, we also use the CPU version of SLIC to generate superpixels for DEL, and we call this variant DEL-C.

Methods	Boundary		Region		Time (s)
	ODS	OIS	ODS	OIS	
DEL-Max	0.703	0.738	0.323	0.389	0.088
DEL-*conv5*	0.667	0.695	0.278	0.343	0.070
DEL-EGB	0.662	0.686	0.305	0.325	0.091
DEL	0.704	0.738	0.326	0.397	0.088
DEL-C	0.715	0.745	0.333	0.402	0.165

Table 1: The ablation study on BSDS500 dataset.

3.1 Ablation Study

BSDS500 [6] is the standard benchmark for image segmentation, oversegmentation, and edge detection. We use this dataset to evaluate the different choices of each DEL's component. The first variant, which is denoted as DEL-Max, replaces the average operation with maximum operation in superpixel pooling while keeping other components as the same with DEL. The second variant, DEL-*conv5*, only uses the final convolutional layer (*conv5*) of VGG16 net. The third variant, DEL-EGB, applies the merging strategy of EGB by viewing each superpixel as a node in the graph partitioning problem.

The evaluation results are summarized in Table 1. Compared with the original DEL, these variants achieve worse performance. It demonstrates the initial choices of the DEL's components are reasonable. For example, our proposed network architecture in DEL can capture both fine-level and coarse-level information, while the naive design of DEL-*conv5* only uses coarse-level information. Thus DEL is much better than DEL-*conv5*. Moreover, EGB seems to be useless for superpixel merging. Maximum pooling is slightly worse than average pooling. It conforms to our intuition that maximum and average pooling usually have similar effects.

3.2 Evaluation on BSDS500 Dataset

Since ODS F-measure is the most important metric for segmentation, we show the ODS F-measure *vs.* running time in Figure 3. Although our proposed DEL does not achieve the best performance, it achieves a good trade-off between efficiency and effectiveness. Among these methods, the fastest one is HFS [8] which can run at real time. However, it suffers poor performance, especially for the region evaluation metric. Thus it can not satisfy today's vision tasks despite its high speed. SLIC [9] and GPU-SLIC [10] seem to struggle on image segmentation. It fits our intuition that over-

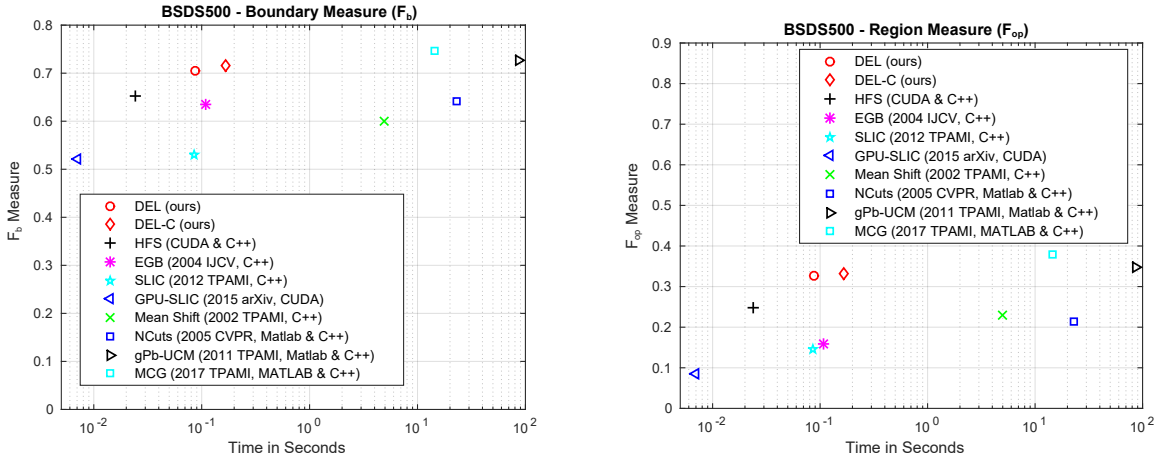

Figure 3: The evaluation results on BSDS500 dataset. **Left:** Boundary measure. **Right:** Region measure.

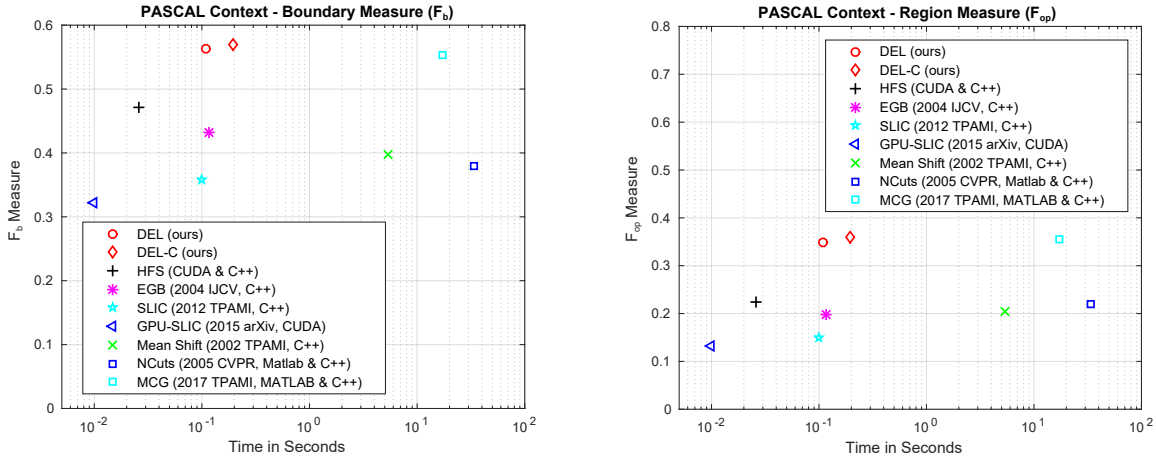

Figure 4: The evaluation results on PASCAL Context dataset. **Left:** Boundary measure. **Right:** Region measure.

segmentation methods are not suitable for image segmentation. The improvement from GPU-SLIC/SLIC to DEL/DEL-C demonstrates the effectiveness of our deep embedding feature learning paradigm. It is interesting to find that GPU-SLIC performs slightly worse than SLIC, and DEL also performs slightly worse than DEL-C. But we choose DEL as the default setting because of the efficiency of GPU-SLIC despite its poor performance. Replacing SLIC with more accurate superpixel generation methods may lead to better performance. DEL provides a converter from superpixels to segmentation. New superpixel techniques will benefit image segmentation in this way. Although MCG [1] achieves accurate results, their low speeds limit their application in many vision tasks. Note that there is no straightforward GPU implementation of MCG, because MCG is not a parallelizable algorithm.

The numeric comparison is summarized in Table 2. The ODS F_b and F_{op} of DEL is 5.2% and 7.7% higher than HFS, respectively. For speed, HFS achieves 41.7fps compared with the 11.4fps of DEL. The accuracy improvement from HFS to DEL is important for many applications. Compared with EGB, DEL achieves better performance both in accuracy and speed. DEL can generate comparable results with state-of-the-art performance, but is much faster. Thus DEL achieves a good trade-off between effectiveness and efficiency. This makes DEL suitable for many high-level vision tasks. We display some qualitative comparisons in Figure 6. We can see that DEL can adapt to complex scenarios and produce more accurate and regular segmented regions.

Methods	Boundary		Region		Time (s)
	ODS	OIS	ODS	OIS	
HFS	0.652	0.686	0.249	0.272	0.024
EGB	0.636	0.674	0.158	0.240	0.108
SLIC	0.529	0.565	0.146	0.182	0.085
GPU-SLIC	0.522	0.547	0.085	0.132	0.007
MShift	0.601	0.644	0.229	0.292	4.95
NCuts	0.641	0.674	0.213	0.270	23.2
gPb-UCM	0.726	0.760	0.348	0.385	86.4
MCG	0.747	0.779	0.380	0.433	14.5
DEL	0.704	0.738	0.326	0.397	0.088
DEL-C	0.715	0.745	0.333	0.402	0.165

Table 2: The evaluation results on BSDS500 dataset.

3.3 Evaluation on PASCAL Context Dataset

PASCAL Context dataset [11] contains 540 categories for semantic segmentation. Due to the pixel-wise labeling of the

Methods	Boundary		Region		Time (s)
	ODS	OIS	ODS	OIS	
HFS	0.472	0.495	0.223	0.231	0.026
EGB	0.432	0.454	0.198	0.203	0.116
SLIC	0.359	0.409	0.149	0.160	0.099
GPU-SLIC	0.322	0.340	0.133	0.157	0.010
MShift	0.397	0.406	0.204	0.214	5.32
NCuts	0.380	0.429	0.219	0.285	33.4
MCG	0.554	0.609	0.356	0.419	17.05
DEL	0.563	0.623	0.349	0.420	0.108
DEL-C	0.570	0.631	0.359	0.429	0.193

Table 3: The evaluation results on PASCAL Context dataset.

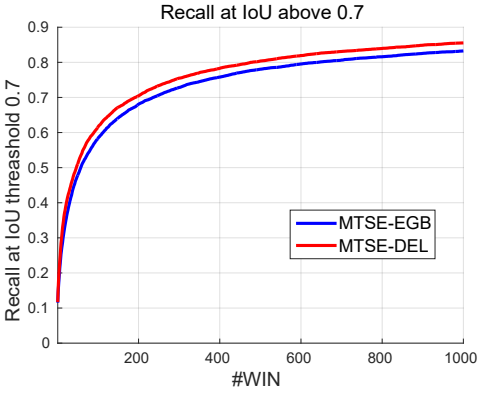

Figure 5: The evaluation of object proposals on PASCAL VOC2007 dataset.

whole image, it can be used to evaluate image segmentation methods. The semantic labeling is converted to ground truth segmentation regions by connectivity labeling. We train our model on the *trainval* set and test on the *test* set. Since there are more test images, this dataset is more challenging than BSDS500.

We summarize the evaluation results in Figure 4. DEL and DEL-C achieve better performance than MCG. Moreover, DEL is about 160 times faster than MCG. One can see that DEL has a good trade-off between accuracy and runtime. We list numeric results in Table 3. DEL is 9.1% and 12.6% higher than HFS on boundary metric and region metric, respectively. This indicates that our learned deep features are more effective than the hand-crafted features used in HFS. Thus this work is a good start to adopt deep features for generic image segmentation.

3.4 Object Proposal Generation

Object proposal generation is necessary for a series of mid-level and high-level vision tasks such as object detection [24] and instance semantic segmentation [25]. Many proposal generation algorithms have been presented, and a considerable portion of these methods are based on image segmentation. To evaluate our proposed DEL in practical applications, we apply it to object proposal generation. MTSE [26] uses the segmented regions of EGB to refine the locations of existing proposals generated by other proposal generation methods. As shown in [26], the BING algorithm [27] has the most significant improvement in performance when MTSE is used as a post-processing step. Thus we replace EGB in MTSE with our DEL to refine the bounding boxes produced by BING. We show the detection recall with IoU overlap 0.7 versus the number of proposals in Figure 5. One can see that MTSE has significant improvement with our DEL segmentation. More experiments of applications are out of scope of this paper, but the proposal evaluation demonstrates the effectiveness of DEL in practical applications.

Figure 6: Some qualitative comparisons. The first column displays original images from BSDS500 dataset. The last four columns show the results generated by EGB, HFS, MCG, and our DEL method, respectively.

4 Conclusion

In this paper, we propose a deep learning based image segmentation algorithm. Specifically, We first use the fast SLIC algorithm to generate superpixels of an input image. Then, the deep embedding feature space that encodes high-level and low-level representation of each superpixel is learned. We propose a similarity metric to convert the learned embedding vector to a similarity value. A simple superpixel merging is performed to obtain perceptual regions according to the similarity values. Our proposed DEL method achieves a good trade-off between efficiency and effectiveness. It makes DEL have the potential to be applied to many vision tasks. Applying DEL to object proposal generation, the quality of generated proposals is significantly improved. In the future, we plan to explore DEL in other applications such as [1; 3; 4].

Acknowledgments

This research was supported by NSFC (NO. 61620106008, 61572264), Huawei Innovation Research Program, and Fundamental Research Funds for the Central Universities.

References

[1] J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE TPAMI*, vol. 39, no. 1, pp. 128–140, 2017.

[2] Z. Zhang, Y. Liu, X. Chen, Y. Zhu, M.-M. Cheng, V. Saligrama, and P. H. Torr, "Sequential optimization for efficient high-quality object proposal generation," *IEEE TPAMI*, 2017.

[3] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *IEEE ICCV*. IEEE, 2011, pp. 1323–1330.

[4] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *CVPR*, 2013, pp. 923–930.

[5] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE TPAMI*, vol. 35, no. 8, pp. 1915–1929, 2013.

[6] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE TPAMI*, vol. 33, no. 5, pp. 898–916, 2011.

[7] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, pp. 167–181, 2004.

[8] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, and Z. Tu, "HFS: Hierarchical feature selection for efficient image segmentation," in *ECCV*. Springer, 2016, pp. 867–882.

[9] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE TPAMI*, vol. 34, no. 11, pp. 2274–2282, 2012.

[10] C. Y. Ren, V. A. Prisacariu, and I. D. Reid, "gSLICr: SLIC superpixels at over 250hz," *arXiv preprint arXiv:1509.04232*, 2015.

[11] 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 *IEEE CVPR*, 2014, pp. 891–898.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," 2007.

[13] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE TPAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[14] D. Comaniciu and P. Meer, "Mean Shift: A robust approach toward feature space analysis," *IEEE TPAMI*, vol. 24, no. 5, pp. 603–619, 2002.

[15] Z. Ren and G. Shakhnarovich, "Image segmentation by cascaded region agglomeration," in *IEEE CVPR*, 2013, pp. 2011–2018.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[17] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *IEEE CVPR*. IEEE, 2017, pp. 3000–3009.

[18] Y. Liu, M.-M. Cheng, J. Bian, L. Zhang, P.-T. Jiang, and Y. Cao, "Semantic edge detection with diverse deep supervision," *arXiv preprint arXiv:1804.02864*, 2018.

[19] 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.

[20] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking wider to see better," in *ICLR*, 2016.

[21] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *IEEE ICCV*. IEEE, 2011, pp. 991–998.

[22] J. Pont-Tuset and F. Marques, "Supervised evaluation of image segmentation and object proposal techniques," *IEEE TPAMI*, vol. 38, no. 7, pp. 1465–1478, 2016.

[23] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *IEEE CVPR*, vol. 2. IEEE, 2005, pp. 1124–1131.

[24] R. Girshick, "Fast R-CNN," in *IEEE ICCV*, 2015, pp. 1440–1448.

[25] A. Arnab and P. H. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *IEEE CVPR*, 2017, pp. 441–450.

[26] X. Chen, H. Ma, X. Wang, and Z. Zhao, "Improving object proposals with multi-thresholding straddling expansion," in *IEEE CVPR*, 2015, pp. 2587–2595.

[27] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *IEEE CVPR*, 2014, pp. 3286–3293.