Pattern Recognition Letters 140 (2020) 135–142

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

End-to-end trainable network for superpixel and image segmentation

Kai Wang, Liang Li[*], Jiawan Zhang

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China

A R T I C L E   I N F O

Article history:
Received 5 January 2020
Revised 7 August 2020
Accepted 11 September 2020
Available online 19 September 2020

MSC:
41A05
41A10
65D05
65D17

Keywords:
Image segmentation
Superpixel
Deep learning

A B S T R A C T

Image segmentation and superpixel generation have been studied for many years, and they are still active research topics in computer vision. Although many advanced computer vision algorithms have been used for image segmentation and superpixel generation, there is no end-to-end trainable algorithm that generates superpixels and segment images simultaneously. We propose an end-to-end trainable network to solve this problem. We train a differentiable clustering algorithm module to produce accurate superpixels. Based on the generated superpixels, the superpixel pooling operation is performed to obtain superpixel features, and then we calculate the similarity of two adjacent superpixels. If the similarity is greater than the preset threshold, we merge the two superpixels. Finally, we get the segmented image. We conduct our experiments in the BSDS500 dataset and get good results.

1. Introduction

As a key component of many different computer vision tasks, *image segmentation* intends to divide an image into large perceptual regions, where the pixels within each region typically belong to the same visual object with minor feature differences. It has been widely used in object proposal generation [2,3], object detection/recognition [4,5] and other fields.

Different from large perceptual regions resulting from image segmentation, *superpixel segmentation* performs an over-segmentation on input image. It segments an image into small, regular, and compact regions, which are often composed of pixels having similar spatial position, texture, color, brightness, *etc.* Meanwhile, it preserves the salient features of a pixel-based representation. Compared to image segmentation, superpixel usually has strong boundary coherence and the produced segments can be easy to control. Due to the efficiency in both representation and computation, superpixels have been widely used in computer vision algorithms, such as object detection [6,7], semantic segmentation [8–11], tracking [12–14], and saliency estimates [15–17].

In recent years, people attempt to use superpixels as a reasonable start for image segmentation, *e.g.* [18], to reduce the computational complexity. However, the interreaction of these two kinds of segmentation is seldom considered. On one hand, superpixel indeed can be viewed as a good start for segmentation due to its good adherence to strong boundaries; On the other hand, results of image segmentation can also provide clues for superpixel generation. Consider performing superpixel segmentation on images with similar foreground and background color for example, it is difficult for algorithm to determine the boundary with only local, low-level pixel features. By integrating object-level features into superpixel segmentation, it is supposed to achieve better result.

In this paper, we try to explore the mutual promotion between image segmentation and superpixel segmentation. Specifically, we propose an end-to-end trainable network which can generate superpixels and perform image segmentation simultaneously. By combining these two tasks under the same neural network framework, we try to demonstrate they promote each other. The architecture of our network is shown in Fig. 1. We use a fully convolutional network and the iterative differentiable clustering algorithm [19] to obtain superpixels. Next, we adopt the superpixel pooling layer [20] to get the superpixel features, with which the similarity between adjacent superpixels can be calculated. If the similarity is greater than the preset threshold, we merge them according to a simple procedure to get object segments. Note that since superpixel does not contain any semantic information while object segmentation does, we get them using different features (shown in Fig. 1 using light blue- and light pink-rectangle). We train the network using BSDS500 segmentation benchmark dataset [21] and compare our result with state-of-the-art ones. Experiment

[*] Corresponding author.
E-mail address: liangli@tju.edu.cn (L. Li).

https://doi.org/10.1016/j.patrec.2020.09.016
0167-8655/© 2020 Elsevier B.V. All rights reserved.

results show our network can produce boundary-adherent superpixels and semantic meaningful objects.

The proposed network has the following characteristics:

- Our network is end-to-end trainable and can be easily assembled into other deep network structures for subsequent applications.
- Our network can generate superpixels and get segmentation results, which is more efficient. The proposed algorithm has excellent performance and has higher precision than existing algorithms.

2. Related work

Since Ren and Malik proposed the concept of superpixel in 2003 [22], researchers have contributed to the field. The simple linear iterative clustering (SLIC) algorithm [23] converts the color image into the CIELAB color spaces and the 5-dimensional feature vectors in XY coordinates, and then constructs a distance metric for the 5-dimensional feature vector to locally cluster the pixels. The idea is simple and easy to implement. Compared with other superpixel segmentation methods, the SLIC is ideal for speed, compactness and contour retention. However, due to the non-differentiable nature of SLIC, it is difficult to combine it with the deep network. In order to solve this problem, Varun Jampani et al. proposed Superpixel Sampling Network (SSN) [19], which converts the SLIC into a differentiable algorithm by relaxing the nearest neighbor constraint existing in the SLIC, so that the deep network can be used to learn superpixels.

There are many mature image segmentation algorithms. Due to space constraints, we only review some classical algorithms. The graph-based approach represents image segmentation as a graph partitioning problem, such as EGB [24]. Although the EGB algorithm is simple and efficient, the optimal value of the parameters is difficult to determine, over-segmentation and under-segmentation may occur. Fuzzy C-means (FCM) clustering algorithm is a kind of fuzzy clustering algorithm. The unsupervised fuzzy clustering can reduce human intervention and complete segmentation automatically. However, there are many defects, such as the initial value is difficult to determine, sensitive to noise, and so on.

Segmentation by learning the similarity of pixels or superpixels is also a trend. In [25], the authors proposed AffinityNet that predicts semantic affinity between a pair of adjacent image coordinates. The semantic propagation is then realized by random walk with the affinities predicted by AffinityNet. Based on the distance metric between the superpixel descriptor vectors to calculate the superpixel similarity, [26] introduced a new superpixel context descriptor to strengthen the learned characteristics towards better similarity prediction. Image segmentation is then achieved by iteratively merging the most similar superpixel pairs selected using a similarity weighting objective function.

Superpixel pooling proposed in Superpixel Pooling Network (SPN) [20] provides new ideas for extracting features of superpixels. SPN utilizes superpixel segmentation of input image as a pooling layout to reflect low-level image structure for learning and inferring semantic segmentation. The initial annotations generated by SPN are then used to learn another neural network that estimates pixelwise semantic labels. The DEL algorithm [18] also used superpixel pooling operation to extract superpixel features to perform image segmentation and achieved good results. Based on the extracted superpixels, our algorithm also uses the superpixel pooling operation to extract the features of the superpixels to calculate the superpixel similarity.

3. Our approach

As shown in Fig. 1, our approach learns image features from the CNN deep network [27] firstly, then we use an iterative differentiable clustering algorithm module to get superpixels. Next, we calculate the superpixel feature vectors by superpixel pooling layer and we learn the similarity between the adjacent superpixels. Finally, we judge whether the adjacent two superpixels are merged according to the similarity. In this section, we will introduce our approach in detail.

3.1. Online superpixel generation

Superpixel groups similar pixels into an isotropic region, which makes it possible to improve the segmentation quality and efficiency. For example, in [18], the authors use SLIC [23] as a start of image segmentation in consideration of efficiency. Here, different from [18] which emploies an existing superpixel algorithm to do image segmentation, we incorporate superpixel generation as part of our image segmentation network. To do this, we adopt differentiable clustering algorithm module, proposed in [19], to replace the hard pixel-superpixel associations in the SLIC algorithm [23].

Formally, for an image $I \in \mathbb{R}^{n\times 5}$ of n pixels with 5-dimensional $I_p = [x, y, l, a, b]$ features in CIELAB space, we expect to divide it into m small regions considered as m superpixels. Here, we briefly review how to compute the hard pixel-superpixel association map $H = \{1, 2, \dots, m\}^{n\times 1}$ in the SLIC algorithm before describing the soft-associations. Given a uniformly sampled superpixel centers C^0

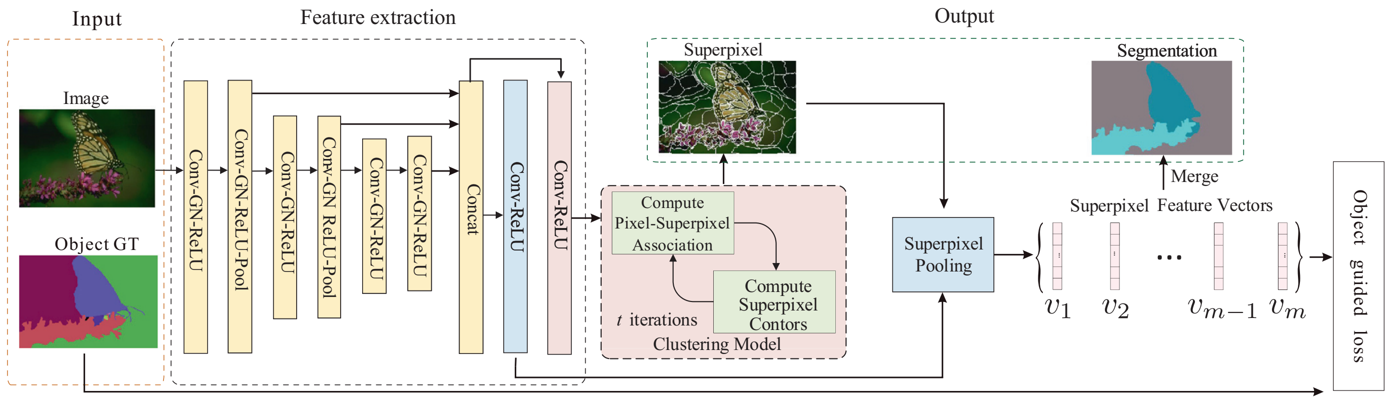

Fig. 1. Pipeline of the proposed algorithm. For a given image, our algorithm generates superpixel and image segmentation simultaneously. The input image is first feeded to a feature-extraction network, which consists of a series of convolutional layers, group normalization (GN) [1] and ReLU nonlinearities. The extracted pixel-level feature are then feeded to the differentiable clustering module to generate superpixels. Superpixel pooling is used to obtain the superpixel feature vectors. To guide the quality of superpixels, we define the loss function based on the object groundtruth. The final image segmentation is achieved by merging superpixels with high similarities.

as initial, the SLIC algorithm computes the new superpixel assignment at each pixel p in each iteration t,

$$H_p^t = \underset{i \in \{1,2,\ldots,m\}}{\operatorname{argmin}} \left\| I_p - C_i^{t-1} \right\|_2, \tag{1}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm of input vector and C_i^{t-1} is the features of superpixel center i which computed by averaging the features of all its belonging pixels in iteration t.

Because of the non-differentiable nearest neighbor assignment evolved in (1), SLIC cannot be directly integrated in neural network. The differentiable clustering algorithm module substitute the hard pixel-superpixel assosiation map H with a soft assosiation $Q \in \mathbb{R}^{n \times m}$, which is differentiable with respect to input features. Similar with original SLIC, it has the following two core steps in each iteration:

1. **Pixel-superpixel association calculation**. The association between pixel p and its neighboring superpixel i in iteration t is calculated as follows,

$$Q_{pi}^t = e^{-\left\| F_p - C_i^{t-1} \right\|_2^2}, \tag{2}$$

where F_p is the deep features of pixel p. In our case, it arises from the feature extraction module of our network. Q_{pi}^t is the (p, i)th entry of Q in iteration t and accounts for the distance between pixel p and superpixel center i.

2. **Superpixel center updating**. The new superpixel clustering center is calculated from a weighted sum of pixel features,

$$C_i^t = \frac{1}{Z_i^t} \sum_p Q_{pi}^t F_p, \tag{3}$$

where Z_i^t denotes a normalization term written as $Z_i^t = \sum_p Q_{pi}^t$.

After iterating these two steps a few times (10 iterations in the experiment), we finally get the soft pixel-superpixel associations $Q \in \mathbb{R}^{n \times m}$. Similar to (1), we calculate the hard association map $H' \in \mathbb{R}^{n \times 1}$ to get the explicit superpixel label for pixel p,

$$H'_p = \underset{i \in \{1,\ldots,m\}}{\operatorname{argmax}} Q_{pi}. \tag{4}$$

Note that, the calculation of such hard-associations is not differentiable. In our algorithm, this step does not participate in back propagation. In the experiment, we find that calculating soft-associations between all pixels and superpixel cluster centers is time-consuming. Similar to SLIC [23], we calculate the distance between each pixel and the surrounding superpixel clustering center, which greatly reduces the calculation time.

3.2. Superpixel similarity

After obtaining the superpixels, we need to measure similarities between them. The feature of superpixels can be calculated using superpixel pooling [20], which is actually an averaging of the belonging pixel features. In our algorithm, while performing superpixel pooling, we use different pixel features from the one used in superpixel generation, as shown in Fig. 1 using light blue-rectangle. The consideration behind is that superpixel contains no semantic information while object segmentation does, so the suitable features for these two tasks would be different. We verify this idea in our experiment by comparing the superpixel and object segmentation results of the proposed algorithm and a variant of that (ours-conv7) quantitatively. Refer to Section 4.1 for more detail.

Formally, by denoting superpixel collection we obtained using $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, we perform superpixel pooling to obtain the superpixel feature vectors $\{v_1, v_2, \ldots, v_m\}$,

$$v_i = \frac{1}{|S_i|} \sum_{p \in S_i} F'_p, \tag{5}$$

where F'_p denotes the feature vector of pixel p used in superpixel pooling and $|\cdot|$ is the cadinality of set (# of pixels in our case).

The similarity between two adjacent superpixel i and j can be calculated by:

$$s_{ij} = \frac{2}{1 + \exp\left(\left\| v_i - v_j \right\|_1 \right)}, \tag{6}$$

where $\|\cdot\|_1$ is the ℓ_1 norm of input vectors. The range of s_{ij} is [0,1]. The larger the value of s_{ij}, the higher the similarity of superpixel i and j. When v_i and v_j are very similar, s_{ij} is close to 1; on the contrary, when v_i and v_j are extremely different, it is close to 0. We decide whether to merge the superpixel i and j based on the similarity s_{ij}.

3.3. Loss function

We assume that the similarity between superpixel pairs in the same segmentation region is greater than the similarity of superpixel pairs in different segmentation regions. Based on the similarity metric defined in (6), we consider the loss function as follows:

$$L = -\sum_{S_i \in S} \sum_{S_j \in \mathcal{R}_i} \left[(1-\alpha) \cdot l_{ij} \cdot \log\left(s_{ij}\right) + \alpha \cdot \left(1 - l_{ij}\right) \cdot \log\left(1 - s_{ij}\right) \right], \tag{7}$$

where $\mathcal{R}_i$ is a set of adjacent superpixels of superpixel S_i, $l_{ij} \in \{0, 1\}$ indicates whether S_i and S_j belongs to the same segmentation region. In practice, l_{ij} is calculated from the obtained superpixel set $\mathcal{S}$ and the groundtruth segmentation masks provided by dataset. In the case S_i and S_j belongs to the same segmentation region, $l_{ij} = 1$; otherwise, $l_{ij} = 0$.

Note that, for different input image, the matrix with (i, j)th entry being defined as l_{ij} is different. So the size of mini-batch during training phase must be set to 1, *i.e.* only one image is fed into the network at a time. The parameter α represents the proportion of superpixel pairs belonging to the same region in groundtruth. It is used to balance the positive and negative samples. By denoting $|Y_+|$ as the number of superpixel pairs belonging to the same region and $|Y|$ as the whole number of superpixel pairs, α is calculated by $\alpha = |Y_+|/|Y|$. Through back propagation, image segmentation also guides superpixel segmentation.

3.4. Superpixel merging

The final image segmentation is obtained by merging similar superpixels. We use the similarity between adjacent superpixels and a preset threshold T to determine whether two adjacent superpixels are merged. Algorithm 1 outlines the calculation steps in superpixel merging.

Algorithm 1 Superpixel merging algorithm.

```
Input: $s$ : similarity;
       $T$ : similarity threshold;
       $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ : superpixels.
Output: Segmentation $\mathcal{S}$.
1: for each $S_i \in \mathcal{S}$ do
2:     Construct adjacent superpixel set $\mathcal{R}_i \subset \mathcal{S}$ of $S_i$;
3:     for each $S_j \in R_i$ do
4:         if $s_{ij} > T$ then
5:             $S_i \leftarrow S_i \cup S_j$, $\mathcal{S} \leftarrow \mathcal{S} \setminus S_j$ ;
6:             Update $\mathcal{R}_i$ ;
7:         end if
8:     end for
9: end for
```

3.5. Network architecture

Fig. 1 shows our network structure. The CNN network used for feature extraction consists of the convolution layer, group normalization (GN) [1] and ReLU activation. We set the number of groups in GN to 8. After the convolution of layer 2 and layer 4, we use the max-pooling to increase the receptive field. The output of layer 4 and 6 is sampled up, and then concatenated with the output of layer 2 to enrich the extracted features. We use the 3×3 convolution filter to set the output channel to 64 per layer.

Note that, we replace the widely used batch normalization (BN) layer with GN in consideration of that the size of each mini-batch must be 1 in our network. Batch normalization (BN) is a milestone technique in the development of deep learning, enabling various networks to train. However, normalizing along the batch dimension introduces problems – BN's error increases rapidly when the batch size becomes smaller, caused by inaccurate batch statistics estimation. In contrast, GN divides the channels into groups and computes within each group the mean and variance for normalization. GN's computation is independent of batch sizes, and its accuracy is stable in a wide range of batch sizes. In the experiment, we also compare the results of using BN and GN as normalization respectively.

In multi-task learning, different levels of tasks require different image features, like UPerNet [28]. For the two different levels tasks, *i.e.* superpixel generation and image segmentation, we further perform convolution operations on the image features obtained in the previous step to obtain different feature vectors to meet the needs of different tasks. Specifically, for the superpixel generation task, we use a convolutional layer with a kernel size of 3×3 to obtain the 30-channel feature vectors. For the image segmentation task, we first perform the 3×3 convolution operation with 256 output channels, and then use the 1×1 convolution kernel to obtain the 64-channel feature vectors. As shown in Fig. 1, we input the obtained feature vectors into the subsequent differentiable clustering algorithm module and superpixel-pooling layer respectively, and then use the proposed corresponding loss function to train the network.

3.6. Implementation details

We implement our network based on Caffe [29], which is a very efficient deep learning framework and widely used in both academia and industry. All codes are written using C++ and Python wrapper of Caffe.

For superpixel generation, like in the original SLIC algorithm, we enforce spatial connectivity across pixels inside each superpixel cluster after (4). This is accomplished by merging the superpixels smaller than certain threshold with the surrounding ones and then assigning a unique cluster ID for each spatially-connected component. For image segmentation, we enforce spatial connectivity after merging superpixels. Note that the operation of enforcing spatial connectivity is not differentiable, we only take it as post-processing and do not add it to neural network.

We use the BSDS500 dataset [21], which has been widely used in the image segmentation community, to server as our training data. Due to the small number of training samples in BSDS500, data augmentation is necessory for training. We treat each groundtruth as a separate sample, *i.e.* for each pair of image and groundtruth, we feed it to the network as a training sample. By using this manner, we get 1633 training/validation pairs and 1063 testing pairs in total. Beside, we adopt 2 common strategies to achieve data augmentation, *i.e.*, flipping and cropping. Specifically, during training phase, we flip images left and right, randomly clip the image into image blocks of size 201×201, to perform data augmentation. Adam [30] is adopted to optimize our network. The basic learning rate is set to 1e-5, and the number of generated superpixels is set to 100. The momentum is set to 0.99 to achieve stable optimization on relatively small-scale data, as suggested in FCN [31]. As described in Section 3.5, the size of each mini-batch is set to 1.

We perform 500K iterations to train the deep learning model and select the final training model based on the accuracy of the verification.

4. Experiments

In this section, we conduct experiments on the widely used BSDS500 dataset [21] to evaluate our approach. The BSDS500 dataset consists of 500 images, including 200 for trainning, 100 for validating, and 200 for testing. It has become a standard benchmark for image segmentation, over-segmentation and edge detection.

Image segmentation and superpixel segmentation are important directions in the field of computer vision, and there are already many publicly available evaluation benchmarks. In our experiments, we use the Boundary F-measure (BF), Probabilistic Rand Index (PRI) and Global Consistency Error (GCE) as our main indicators of segmentation, and use the BF, Boundary Recall (BR), undersegmentation error (UE) and compactness as our main indicators of superpixel. We report our main indicators at the optimal dataset scale (ODS). We argue that ODS is more practical than optimal image scale (OIS), because for real applications there are no human-annotated segmentations. We select optimal parameters based on overall performance at the scale of the whole dataset. The higher the scores of BF, BR, PRI and compactness, the better the results. The lower the GCE score and UE score, the better the results.

To assess the effectiveness and efficiency of our algorithm, we will compare with some of the most advanced segmentation algorithms, such as SSN [19], FLIC [32], SLIC [23], LSC [33], WT [34], DEL [18], SFFCM [35], align-hier [36], EGB [24]. Among them, SSN, FLIC, SLIC, LSC, and WT are superpixel algorithms. DEL, SFFCM, align-hier, and EGB are image segmentation algorithms.

4.1. Hyper-parameter analysis

We use the network structure shown in Fig. 1 as the main model that we name as ours-GN8, and use the BSDS500 dataset to evaluate the different choices of each component in the network. We compare the performance in terms of superpixel generation and image segmentation of 4 variants of ours-GN8, which are listed as follows,

1. ours-BN: Substituting group normalization (GN) operation with batch normalization (BN);
2. ours-GN32: During GN, setting the group number to 32 instead of 8 in ours-GN8;
3. ours-conv7: Using the same feature, which is obtained from the 7th convolutional layer (`conv7`), for both superpixel and image segmentation;
4. ours-w/o-concat: Discarding the shortcuts `conv2` $\rightarrow$ `concat1` and `conv4` $\rightarrow$ `concat1` (*i.e.* not concating features from `conv2` and `conv4`).

We evaluate the results from image segmentation and superpixel generation.

As can be seen from Tables 1 and 2, the original network works best compared with other variants, indicating the rationality of the components we selected. GN addresses the effect of BN on the batch size dependency. For small batch size, GN can get better effect. But when there are too many groups, the effect decreases. According to our experience, the shallow network includes more

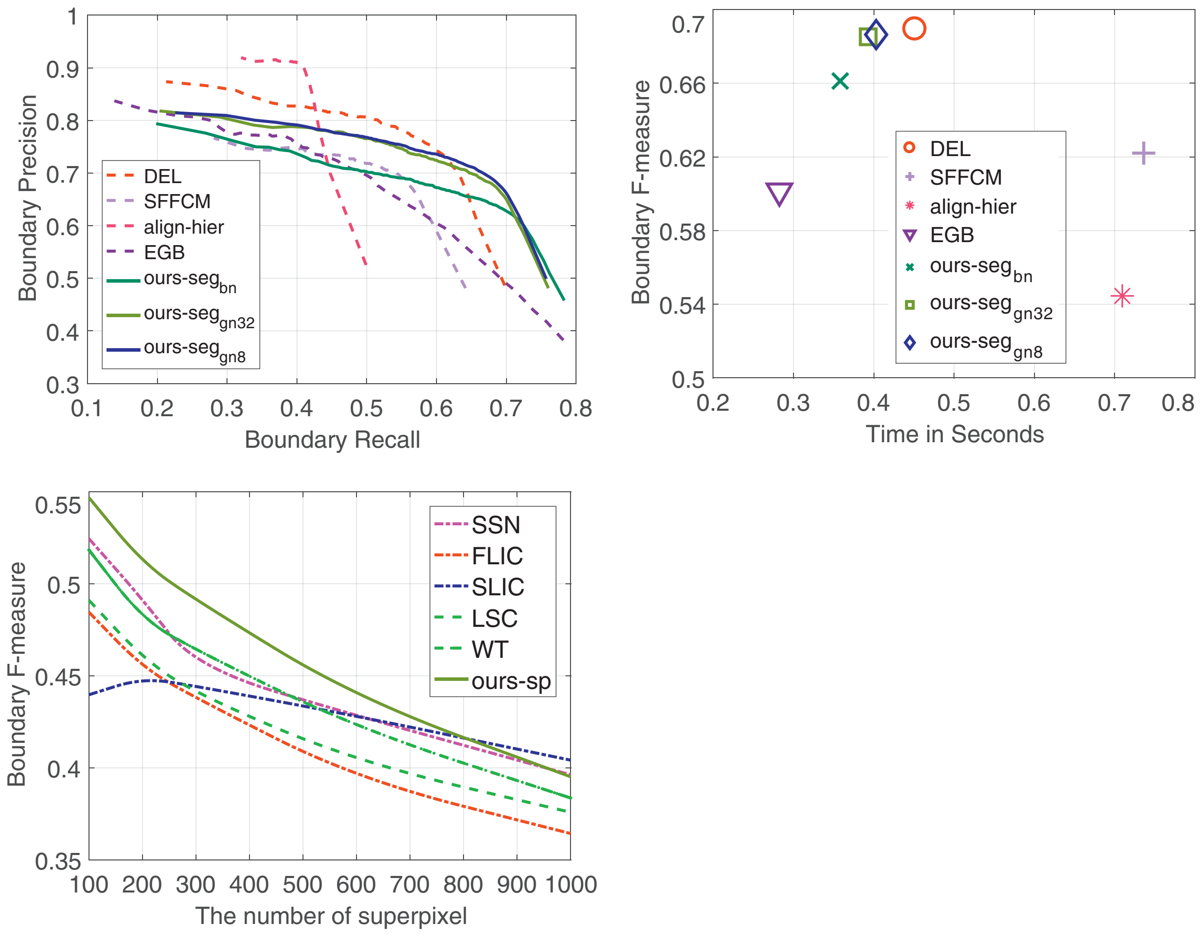

Fig. 2. Results on BSDS500 dataset. **Left**: Boundary P-R on segmentation, **Middle**: Boundary and Time on segmentation, **Right**:Boundary on superpixel.

Table 1
The performance of superpixel generation of 4 variants.

Methods	BF(↑)	BR(↑)	UE(↓)	compactness(↑)
ours-BN	0.547	0.884	0.068	0.373
ours-GN32	0.546	0.897	0.066	0.376
ours-conv7	0.521	0.812	0.094	0.413
ours-w/o-concat	0.521	0.895	0.071	0.340
ours-GN8	0.547	0.918	0.065	0.316

Table 2
The performance of image segmentation of 4 variants.

Methods	BF(↑)	PRI(↑)	GCE(↓)
ours-BN	0.661	0.807	0.146
ours-GN32	0.685	0.820	0.171
ours-conv7	0.492	0.714	0.088
ours-w/o-concat	0.560	0.817	0.182
ours-GN8	0.686	0.822	0.170

detailed information, the deep network contains more global information, and the original network extracts more features than ours-w/o-concat, so the segmentation result is better. The effect of ours-conv7 decrease significantly, ours-GN8 is much better than ours-conv7, thus verifying that in multi-task learning, different levels of tasks require different image features, like UPerNet [28].

4.2. Evaluation experiments

For superpixel generation and image segmentation, we compare the classic algorithms with these two tasks on BSDS500 dataset. Fig. 2 shows the comparison of our algorithm with other algorithms. The left and the middle one show segmnetation result comparisons between the variants of our algorithm and other ones. It can be seen from the first 2 subfigures of Fig. 2 that our algorithm performs well in image segmentation and is superior to other algorithms on the boundary F-measure and time. EGB (▽) does not work well, although it takes the least amount of time. DEL (○) achieves the best on the boundary F-measure, but it is slower than our algorithm. The last subfigure of Fig. 2 shows the quantitative comparisons between our superpixel and the state-of-the-art ones. It can be seen that the superpixels we obtained achieve the best performance on the boundary F-measure for most configurations. In summary, our algorithm achieves a balance between time and effect.

Quantitative comparison is further detailed in Table 3 and 4 (The top two are highlighted in **bold** and underline, respectively). We generate 100, 300 and 600 superpixels on a 321×481 image. Compared with other superpixel algorithms, although the superpixels we generated are not completely compact, they perform best on the BF and BR, and perform well on UE. In addition, the image segmentation we produced achieves good performance both on the boundary F-measure and on the PRI and GCE. As can be seen from Table 3 and 4, our algorithm performs well in superpixel generation and image segmentation, and can generate results equivalent to those of advanced algorithms. And because our network is end-to-end trainable, this makes our algorithm more suitable for many advanced visual tasks.

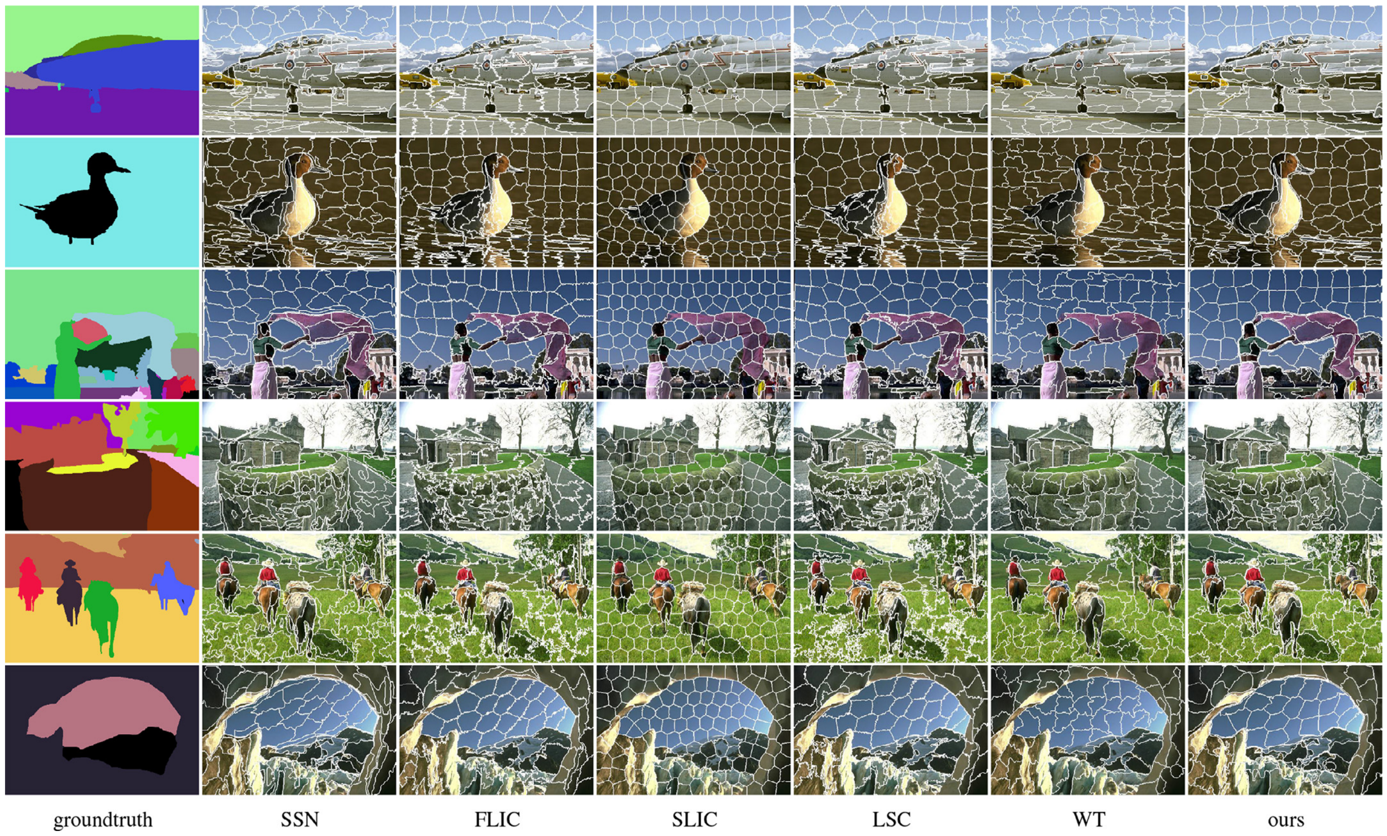

Fig. 3. Superpixel generation. The first column displays groundtruth from BSDS500 dataset. The last six columns show the results generated by SSN, FLIC, SLIC, LSC, WT and our method, respectively.

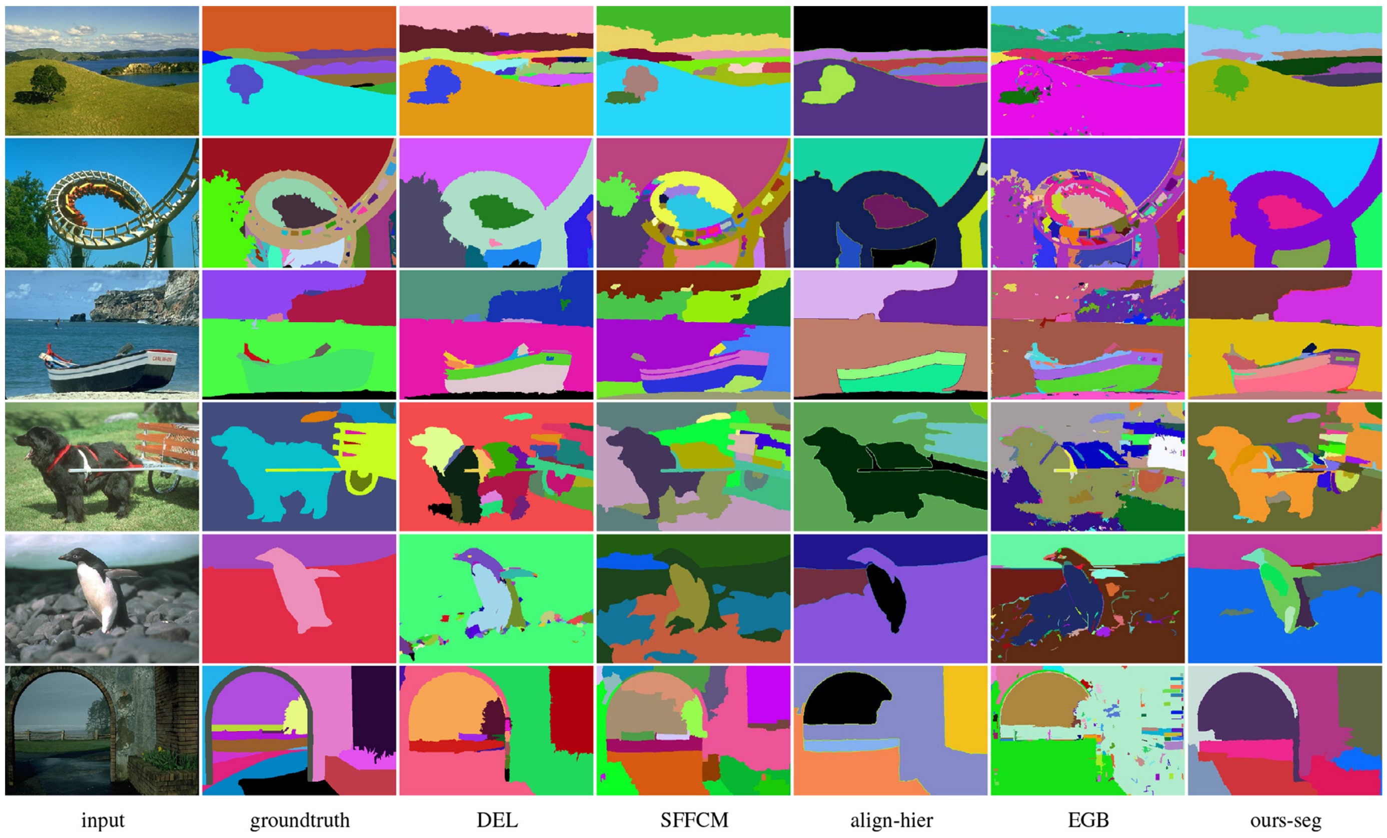

Fig. 4. Image segmentation. The first two columns display original images and groundtruth from BSDS500 dataset. The last five columns show the results generated by DEL, SFFCM, align-hier, EGB and our method, respectively.

Table 3
The evaluation results of superpixel generation.

Methods	N = 100				N = 300				N = 600			
	BF(↑)	BR(↑)	UE(↓)	compactness(↑)	BF(↑)	BR(↑)	UE(↓)	compactness(↑)	BF(↑)	BR(↑)	UE(↓)	compactness(↑)
SSN	*0.524*	*0.911*	**0.060**	0.340	0.454	**0.964**	**0.042**	0.439	*0.423*	**0.985**	**0.035**	0.519
FLIC	0.485	0.845	0.141	0.249	0.427	0.941	0.068	0.280	0.374	*0.980*	0.049	0.352
SLIC	0.440	0.552	0.145	**0.661**	0.457	0.823	0.109	0.442	0.421	0.913	0.079	0.537
LSC	0.491	0.873	0.095	0.288	0.439	*0.950*	0.059	0.414	0.398	0.979	0.045	0.509
WT	0.518	0.837	0.124	*0.438*	*0.464*	0.868	0.076	**0.586**	0.422	0.934	0.060	*0.719*
ours-sp	**0.547**	**0.918**	*0.065*	0.316	**0.491**	0.941	*0.048*	*0.551*	**0.438**	0.955	*0.044*	**0.739**

Table 4
The evaluation results of image segmentation.

Methods	BF(↑)	PRI(↑)	GCE(↓)
DEL	**0.689**	*0.809*	*0.161*
SFFCM	0.622	0.776	0.259
align-hier	0.545	0.738	**0.141**
EGB	0.602	0.763	0.244
ours-seg	*0.686*	**0.821**	0.170

We show some qualitative comparisons in Figs. 3 and 4. As shown in Fig. 3, for superpixel segmentation, the boundaries of the FLIC and LSC results are irregular. The superpixel generated by SLIC is more regular, but the boundary adherence is lacking. Our results can obtain more regular boundaries and better boundary adherence. As shown in Fig. 4, for image segmentation, the results of DEL, SFFCM, and EGB are more fragmented, and the segmentation results of the align-hier method lack some details. Our results are less fragmented and visually closer to groundtruth. In summary, it can be seen that our algorithm can reach a good level in both superpixel segmentation and image segmentation.

5. Conclusion

In this paper, we have proposed an end-to-end trainable network that can generate both superpixel and image segmentation. Specifically, we use a fully convolutional network to extract features of the image and then use the differentiable clustering algorithm module to produce accurate superpixels. The superpixel feature is obtained by using the superpixel pooling operation, and the similarity of two adjacent superpixels is calculated to determine whether to merge or not to obtain the sensing region.

Our proposed algorithm has achieved good performance in both superpixel generation and image segmentation. Besides, since the proposed algorithm is end-to-end trainable, it can be easily integrated into other deep network structures. It makes our algorithm have the potential to be applied to many other vision tasks.

As the future work, we plan to try some other merging algorithm to get the segmentation results. The merging procedure used in our paper is relatively simple and considers only the local similarity. Some other procedures, such normlized cut [37] which adopt a global perspective, is supposed to generate more consistent results. Besides, we also plan to explore the applications of our algorithm in other tasks such as [4,13,14].

Declaration of Competing Interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

[1] Y. Wu, K. He, Group normalization, in: Proc. of ECCV, 2018.
[2] J. Pont-Tuset, P. Arbelaez, J.T. Barron, F. Marques, J. Malik, Multiscale combinatorial grouping for image segmentation and object proposal generation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (1) (2016) 128–140.
[3] J.R. Uijlings, K.E. Van De Sande, T. Gevers, A.W. Smeulders, Selective search for object recognition, Int. J. Comput. Vis. 104 (2) (2013) 154–171.
[4] M. Juneja, A. Vedaldi, C. Jawahar, A. Zisserman, Blocks that shout: distinctive parts for scene classification, in: Proc. of IEEE CVPR, 2013.
[5] P. Kohli, P.H. Torr, et al., Robust higher order potentials for enforcing label consistency, Int. J. Comput. Vis. 82 (3) (2009) 302–324.
[6] G. Shu, A. Dehghan, M. Shah, Improving an object detector and extracting regions using superpixels, in: Proc. of IEEE CVPR, 2013.
[7] J. Yan, Y. Yu, X. Zhu, Z. Lei, S.Z. Li, Object detection by labeling superpixels, in: Proc. of IEEE CVPR, 2015.
[8] S. Gould, J. Rodgers, D. Cohen, G. Elidan, D. Koller, Multi-class segmentation with relative location prior, Int. J. Comput. Vis. 80 (3) (2008) 300–316.
[9] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, P.V. Gehler, Superpixel convolutional networks using bilateral inceptions, in: Proc. of ECCV, 2016.
[10] A. Sharma, O. Tuzel, M.-Y. Liu, Recursive context propagation network for semantic scene labeling, in: Proc. of NeurIPS, 2014.
[11] X. Xie, G. Xie, X. Xu, L. Cui, J. Ren, Automatic image segmentation with superpixels and image-level labels, IEEE Access 7 (2019) 10999–11009.
[12] S. Wang, H. Lu, F. Yang, M.-H. Yang, Superpixel tracking, in: Proc. of ICCV, 2011.
[13] F. Yang, H. Lu, M.-H. Yang, Robust superpixel tracking, IEEE Trans. Image Process. 23 (4) (2014) 1639–1651.
[14] P.-H. Conze, F. Tilquin, M. Lamard, F. Heitz, G. Quellec, Unsupervised learning-based long-term superpixel tracking, Image Vis. Comput. 89 (2019) 289–301.
[15] S. He, R.W. Lau, W. Liu, Z. Huang, Q. Yang, SuperCNN: a superpixelwise convolutional neural network for salient object detection, Int. J. Comput. Vis. 115 (3) (2015) 330–344.
[16] C. Yang, L. Zhang, H. Lu, X. Ruan, M.-H. Yang, Saliency detection via graph-based manifold ranking, in: Proc. of IEEE CVPR, 2013.
[17] W. Zhu, S. Liang, Y. Wei, J. Sun, Saliency optimization from robust background detection, in: Proc. of IEEE CVPR, 2014.
[18] Y. Liu, P.-T. Jiang, V. Petrosyan, S.-J. Li, J. Bian, L. Zhang, M.-M. Cheng, DEL: deep embedding learning for efficient image segmentation, in: Proc. of IJCAI, 2018.
[19] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, J. Kautz, Superpixel sampling networks, in: Proc. of ECCV, 2018.
[20] S. Kwak, S. Hong, B. Han, Weakly supervised semantic segmentation using superpixel pooling network, in: Proc. of AAAI, 2017.
[21] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2010) 898–916.
[22] X. Ren, J. Malik, Learning a classification model for segmentation, in: Proc. of ICCV, 2003.
[23] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, SLIC superpixels compared to state-of-the-art superpixel methods, IEEE Trans. Pattern Anal. Mach. Intell. 34 (11) (2012) 2274–2282.
[24] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vis. 59 (2) (2004) 167–181.
[25] J. Ahn, S. Kwak, Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation, in: Proc. of IEEE CVPR, 2018.
[26] M.S. Chaibou, P.-H. Conze, K. Kalti, M.A. Mahjoub, B. Solaiman, Learning contextual superpixel similarity for consistent image segmentation, Multimed. Tools. Appl. 79 (3–4) (2020) 2601–2627.
[27] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proc. of NeurIPS, 2012.
[28] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, J. Sun, Unified perceptual parsing for scene understanding, in: Proc. of ECCV, 2018.
[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proc. of ACM MM, 2014.

[30] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proc. of ICLR, 2015.
[31] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proc. of IEEE CVPR, 2015.
[32] J. Zhao, B. Ren, Q. Hou, M.-M. Cheng, P.L. Rosin, FLIC: fast linear iterative clustering with active search, in: Proc. of AAAI, 2018.
[33] Z. Li, J. Chen, Superpixel segmentation using linear spectral clustering, in: Proc. of IEEE CVPR, 2015.
[34] Z. Hu, Q. Zou, Q. Li, Watershed superpixel, in: Proc. of ICIP, 2015.
[35] T. Lei, X. Jia, Y. Zhang, S. Liu, H. Meng, A.K. Nandi, Superpixel-based fast fuzzy c-means clustering for color image segmentation, IEEE Trans. Fuzzy Syst. 27 (9) (2019) 1753–1766.
[36] Y. Chen, D. Dai, J. Pont-Tuset, L. Van Gool, Scale-aware alignment of hierarchical image segmentation, in: Proc. of IEEE CVPR, 2016.
[37] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE, Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.