AARHUS
UNIVERSITET

Computer Vision & Machine Learning

Alexandros Iosifidis
@
Department of Electrical and Computer Engineering
Aarhus University

Image segmentation based on saliency

What is visual saliency?

Image segmentation based on saliency

What is visual saliency?

Image segmentation based on saliency

Saliency-based image segmentation:

- Detect the most distinctive regions/object(s) in an image
- Segment (in an accurate manner) the region of the salient object(s)

The above process is (almost) trivial for humans, but not so easy for computers

Saliency-based image segmentation is the first step for numerous higher-level analysis tasks:

- scene analysis
- behavior analysis
- …

Image segmentation based on saliency

Criteria for good saliency-based segmentation:

- Good detection:
  - all salient objects need to be detected
  - background regions need not be selected as salient ones

- High resolution: the resulting “saliency maps” need to be of high resolution and accurately segment out the salient object boundaries

- Computational efficiency: (as a pre-processing step) methods need to be fast

AARHUS UNIVERSITET

Department of Electrical and Computer Engineering

Computer Vision & Machine Learning

Image segmentation based on saliency

Three types of visual saliency problems:

- Fixation prediction (where a human observer is more likely to look?)
- Salient object Segmentation
- Image segmentation based on semantic classes

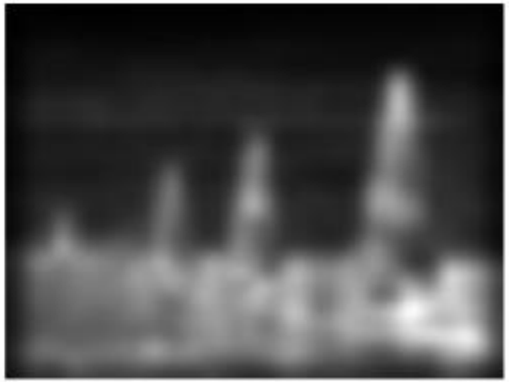
These problems are closely related. We will focus on the salient object segmentation and semantic image segmentation problems.

Image segmentation based on saliency

Original image

Saliency map

Fixation prediction



Segmented image

Super-pixels

Object proposals

Image segmentation based on saliency

Salient object segmentation methods work on four levels:

- Pixels:
  - too computationally costly due to the high number of pixels in high-resolution images
  - pixels (alone) do not convey much information in terms of saliency
- Blocks or patches (regularly sampled image patches):
  - reduce the computational cost, but ignores the shape information of objects
- Regions (obtained by applying image segmentation methods):
  - reduce the computational cost, since the number of all (distinct) image regions is much lower than the number of pixels and blocks
  - using image segmentation based on super-pixels we have a compromise between pixel- and large image region-based saliency detection
- Entire image (usually used by neural network models)

Image segmentation based on saliency

Salient object segmentation methods based on the type of information they exploit:

- Intrinsic:
  - exploit only the given image
  - encode several properties of saliency (saliency cues) defined on one image

- Extrinsic:
  - exploit enriched information (not only included in the given image), such as user annotations, information related to images which are similar to the given image
  - encode saliency cues defined on a set of images or train a saliency model on a training set of images

Image segmentation based on saliency

Block-based segmentation:

- A full-resolution saliency can be obtained by calculating the saliency score of each pixel (at location x):

$$s(x) = \|I_{\mu} - I_{\omega_{hc}}(x)\|^2$$

I_μ is the mean pixel value of the image (e.g. RBG or Lab features) and $I_{\omega hc}$ is a Gaussian blurred version of the input image (e.g. by applying a 5x5 average filter)

image sal.map

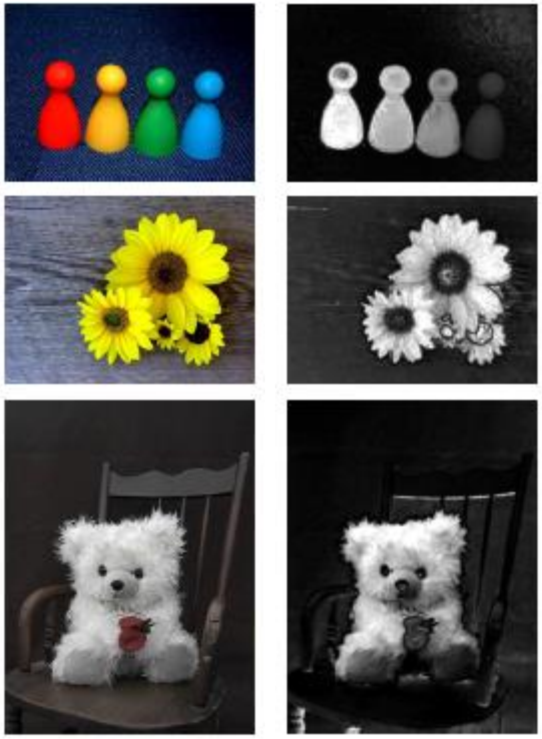


Image segmentation based on saliency

Block-based segmentation:

- In order to handle objects of different sizes an L-layer Gaussian pyramid can be used and the saliency score can be defined as:

$$s(x) = \sum_{l=1}^{L} \sum_{x' \in \mathcal{N}(x)} ||I^{(l)}(x) - I^{(l)}(x')||^2$$

N(x) is a neighboring window centered at x (e.g. 9x9 pixels) and $I^{(l)}$ is the image at the l-th level of the pyramid

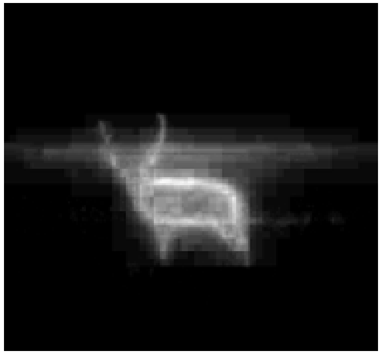


Image segmentation based on saliency

Region-based segmentation:

- The saliency is calculated by expressing various types of saliency cues on small image regions obtained through:
  - Image segmentation
  - Super-pixel calculation

Image segmentation based on saliency

Types of saliency cues used:

- Global regional contrast: a salient region needs to have high contrast with the background regions
- Priors:
  - center prior: people usually center the silent objects in images
  - backgroundness prior: a narrow border of the image is usually considered to belong to the background
  - background connectivity prior: a salient object is much less connected to the image border than objects in the background
  - face prior: faces are usually considered as salient objects
  - objectness prior: locations depicting objects are more likely to be salient than locations not depicting objects
- Spatial distribution prior: the wider a color is distributed in an image, the less likely a salient object contains this color

Simple Linear Iterative Clustering (SLIC)

An adaptation of K-Means algorithm for super-pixel generation with two distinctions:

- Limitations on the search space to a region proportional to the super-pixel size
- Use of weighted distance measure combining color and spatial proximity, while at the same time provides control on the compactness and the size of the super-pixels

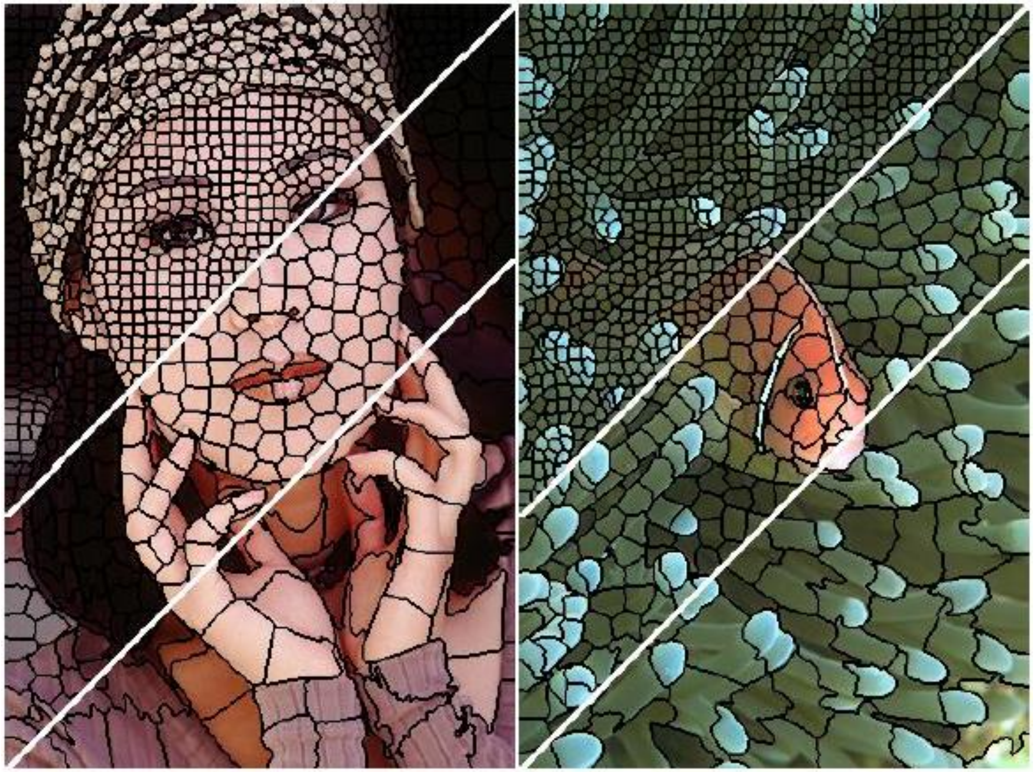


Fig. 1: Images segmented using SLIC into superpixels of size 64, 256, and 1024 pixels (approximately).

Simple Linear Iterative Clustering (SLIC)

Algorithm 1 SLIC superpixel segmentation

/* *Initialization* */
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  /* *Assignment* */
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  /* *Update* */
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Simple Linear Iterative Clustering (SLIC)

Work on the LAB color space augmented with the image locations

Algorithm 1 SLIC superpixel segmentation

/∗ Initialization ∗/
Initialize cluster centers $\boxed{C_k = [l_k, a_k, b_k, x_k, y_k]^T}$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.
repeat
  /∗ Assignment ∗/
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  /∗ Update ∗/
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Simple Linear Iterative Clustering (SLIC)

Work on the LAB color space augmented with the image locations

Initialize the cluster centers in a regular grid

Algorithm 1 SLIC superpixel segmentation

/∗ Initialization ∗/
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  /∗ Assignment ∗/
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  /∗ Update ∗/
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Simple Linear Iterative Clustering (SLIC)

Work on the LAB color space augmented with the image locations

Initialize the cluster centers in a regular grid

Slightly update the cluster centers in order to avoid a cluster center falling on an edge

Algorithm 1 SLIC superpixel segmentation

/∗ Initialization ∗/
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  /∗ Assignment ∗/
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  /∗ Update ∗/
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Simple Linear Iterative Clustering (SLIC)

Work on the LAB color space augmented with the image locations

Initialize the cluster centers in a regular grid

Slightly update the cluster centers in order to avoid a cluster center falling on an edge

Since we want to obtain regularly-spaced clusters, restrict the search space in an area of 2S x 2S from each cluster center

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  / Assignment */*
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  / Update */*
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Simple Linear Iterative Clustering (SLIC)

Work on the LAB color space augmented with the image locations

Initialize the cluster centers in a regular grid

Slightly update the cluster centers in order to avoid a cluster center falling on an edge

Since we want to obtain regularly-spaced clusters, restrict the search space in an area of 2S x 2S from each cluster center

Update the cluster centers and compute the new error

Algorithm 1 SLIC superpixel segmentation

```
/* Initialization */
Initialize cluster centers C_k = [l_k, a_k, b_k, x_k, y_k]^T by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3 × 3 neighborhood.
Set label l(i) = −1 for each pixel i.
Set distance d(i) = ∞ for each pixel i.

repeat
  /* Assignment */
  for each cluster center C_k do
    for each pixel i in a 2S × 2S region around C_k do
      Compute the distance D between C_k and i.
      if D < d(i) then
        set d(i) = D
        set l(i) = k
      end if
    end for
  end for
  /* Update */
  Compute new cluster centers.
  Compute residual error E.
until E ≤ threshold
```

Simple Linear Iterative Clustering (SLIC)

Because the cluster centers' representations are obtained by two sub-vectors of different nature (lab and xy), we use the following distance measure:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$
$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$
$$D' = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}.$$

where N_s and N_c are the maximum distances within a cluster. We set N_s = S and N_c = m (a hyper-parameter, set in the range [1,40])

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  / Assignment */*
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  / Update */*
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Image segmentation based on saliency

Saliency as sparse noise in the image:

- Basic assumption is that non-salient/background regions can be explained by using a low-rank matrix, while the salient regions form the details of the image (in the same sense as explaining the shape of a class in a feature space using only few principal components).

Image segmentation based on saliency

Steps:

- Image over-segmentation (by applying a super-pixel calculation algorithm)
- Representation of each region/super-pixel in a D-dimensional feature space:
  - mean RBG color values, Gabor filter values, etc
- Stacking all representations of super-pixels in a matrix $\mathbf{F} = [f_1, f_2, ..., f_N]$. $\mathbf{F} \in \mathbb{R}^{D \times N}$
- Try to reconstruct **F** using a low-rank matrix **L** and a sparse matrix **S**:

$$(\mathbf{L}^*, \mathbf{S}^*) = arg \min_{\mathbf{L},\mathbf{S}} (rank(\mathbf{L}) + \lambda \parallel \mathbf{S} \parallel_0) \quad s.t. \quad \mathbf{F} = \mathbf{L} + \mathbf{S}$$

the above optimization problem is approximated by:

$$(\mathbf{L}^*, \mathbf{S}^*) = arg \min_{\mathbf{L},\mathbf{S}} (\parallel \mathbf{L} \parallel_* + \lambda \parallel \mathbf{S} \parallel_1) \quad s.t. \quad \mathbf{F} = \mathbf{L} + \mathbf{S}$$

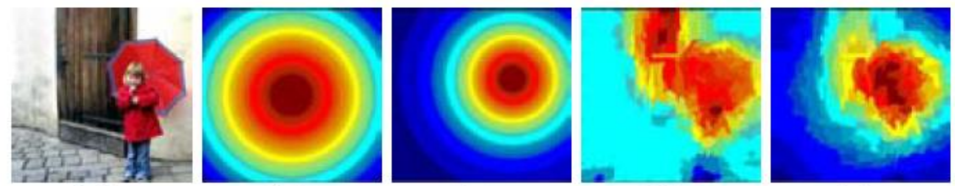
Nuclear norm of **L**

given by tr(sqrt(**L***__L__))

Image segmentation based on saliency

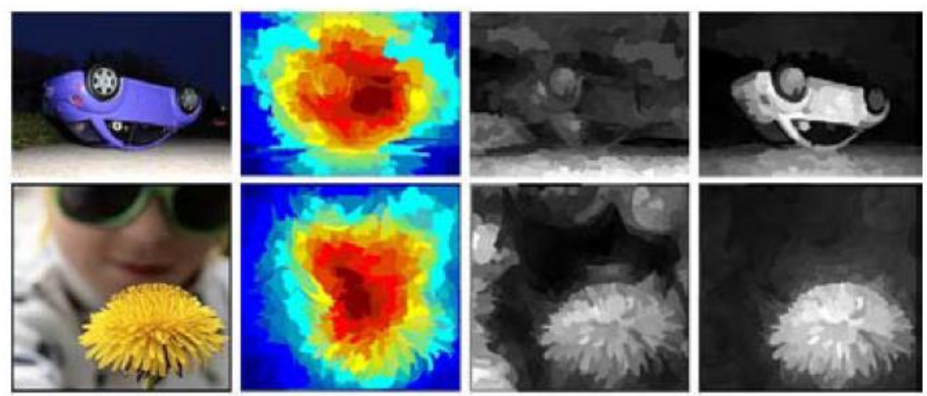
Incorporation of high-level saliency priors:

- Element-wise multiplication of various saliency prior maps.



(a) (b) (c) (d) (e)

Figure 4. Example of high-level prior maps. (a) original image, (b) location prior map, (c) prior maps generated by face detection, (d) color prior map, (e) final fused prior map.



(a) Input (b) Prior map (c) w/o prior (d) w/ prior

Graph-based Saliency Detection

Steps:

- Image over-segmentation (by applying a super-pixel calculation algorithm)
- Representation of each region/super-pixel in a D-dimensional feature space:
  - mean RBG color values, Gabor filter values, etc
- Definition of an undirected graph G(**V**,**E**), where the nodes $\mathbf{V} \in \mathbb{R}^{DxN}$ of the graph are the N super-pixels of the image and the edges **E** connecting them are weighted using a weighting matrix $\mathbf{W} \in \mathbb{R}^{NxN}$ expressing pair-wise similarities between graph nodes.

We will describe three graph-based methods:

- Diffusion-based saliency detection
- Quantum-Cuts
- Probabilistic Saliency Estimation

Graph-based Saliency Detection

Diffusion-based saliency:

- Apply SLIC for defining super-pixels
- Represent each super-pixel using the mean LAB color value within it
- Define an undirected graph G(**V**,**E**), where the nodes **V** $\in \mathbb{R}^{DxN}$ of the graph are the N super-pixels of the image and the edges E connecting them are weighted using a weighting matrix **W** $\in \mathbb{R}^{NxN}$ expressing pair-wise similarities between graph nodes:

$$w_{ij} = e^{-\frac{\|v_i - v_j\|_2}{\sigma^2}}$$

where σ is a hyper-parameter. The diagonal matrix **D** $\in \mathbb{R}^{NxN}$ is given by $D = \text{diag}\{d_{11},\ldots,d_{NN}\}$, where $d_{ii} = \sum_j w_{ij}$

- Saliency map y is obtained by:

$$y = A^{-1} s$$

Diffusion matrix

Seed vector

Graph-based Saliency Detection

Diffusion matrix **A** definition:

- Unnormalized Laplacian matrix: $\mathbf{A} = \mathbf{L} = \mathbf{D} - \mathbf{W}$
- Normalized Laplacian matrix: $\mathbf{A} = \mathbf{L} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{W})$

The diffusion matrix is positive semi-definite and can be written as $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1,\ldots,\lambda_N)$, $0 = \lambda_1 \leq \ldots \leq \lambda_N$ and $\mathbf{U} = [\mathbf{u}_1,\ldots,\mathbf{u}_N]$.

We discard the zero eigenvalue λ_1 and the corresponding eigenvector $\mathbf{u}_1$, since $\mathbf{u}_1 = \mathbf{1}$.

Each eigen-vector is evaluated w.r.t. its discriminability based on its variance:

$$dc(u_l) = \begin{cases} 0, & var(u_l) < v \\ 1, & else \end{cases},$$

$$\widetilde{DC} = diag\{dc(u_2), \ldots, dc(u_r)\}$$

Graph-based Saliency Detection

Each eigen-vector is evaluated w.r.t. its discriminability based on its variance:

$$dc(u_l) = \begin{cases} 0, & var(u_l) < v \\ 1, & else \end{cases},$$
$$\widetilde{DC} = diag\{dc(u_2), \ldots, dc(u_r)\}$$

Then, we re-construct the diffusion matrix as follows:

$$\widetilde{U} = [u_2, \ldots, u_r];$$
$$\widetilde{\Lambda}^{-1} = diag\{\lambda_2^{-1}, \ldots, \lambda_r^{-1}\}$$
$$\widetilde{A}^{-1} = \widetilde{U}\widetilde{\Lambda}^{-1}\widetilde{DC}\widetilde{U}^T$$

Then, the seed vector is defined as: $\tilde{s} = \widetilde{A}^{-1}x$, where $x \in \mathbb{R}^N$ and $x_i = 1$ if v_i is a non-border node/super-pixel and $x_i = 0$, otherwise.

Finally, $y = \widetilde{A}^{-1}\tilde{s} = (\widetilde{A}^{-1})^2 x$

Graph-based Saliency Detection

Diffusion based Saliency detection

Algorithm 1 Promoted Diffusion-Based Salient Object Detection

Input: An image on which to detect the salient object.

1: Segment the input image into superpixels, use the superpixels as nodes, connect border nodes to each other and connect close nodes to construct a graph G, and compute its degree matrix D and weight matrix W.
2: Compute $L_{rw} = D^{-1}(D - W)$ and its eigenvalues and eigenvectors.
3: Estimate the eigengap of L_{rw} by Eq. 9, discard the first constant eigenvector and the eigenvectors after the eigengap.
4: Re-weight the remaining eigenvectors by discriminability as computed by Eq. 10.
5: Form the re-synthesized diffusion matrix $\widetilde{A}^{-1}$ by Eq. 11 and compute the seed vector $\widetilde{s}$ by Eq. 13.
6: Compute the final saliency vector y by Eq. 14.

Output: The saliency vector y representing the saliency value of each superpixel.

Graph-based Saliency Detection

Diffusion based Saliency detection

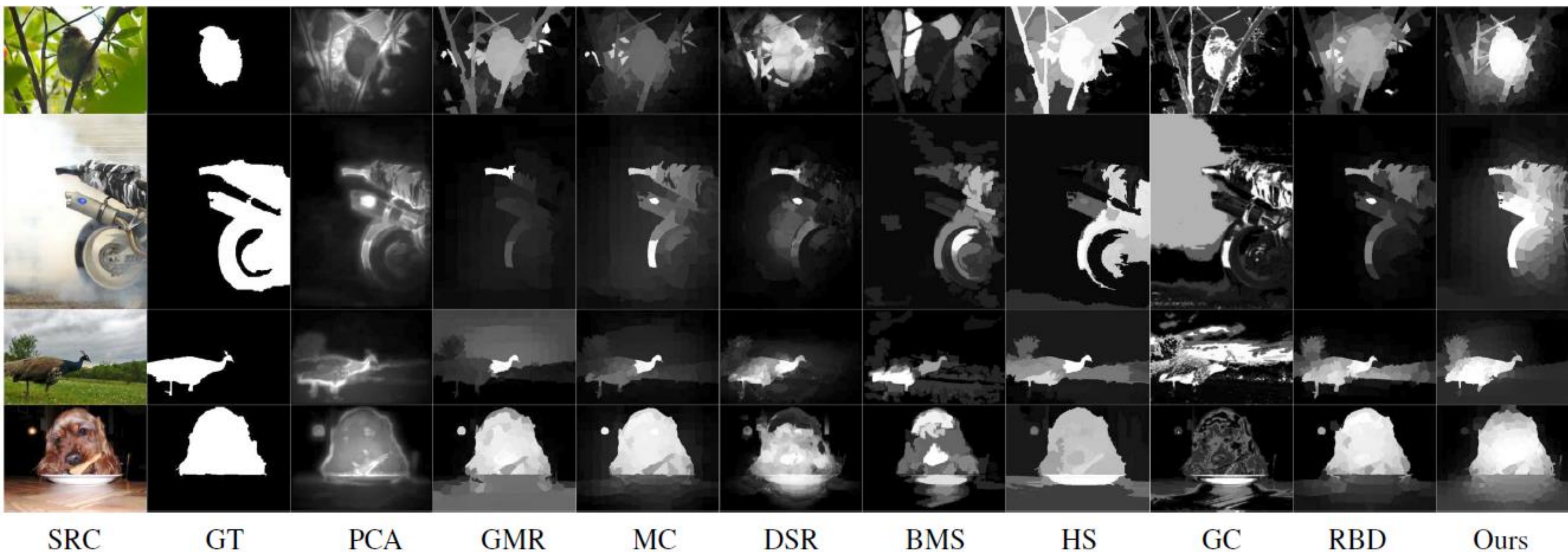
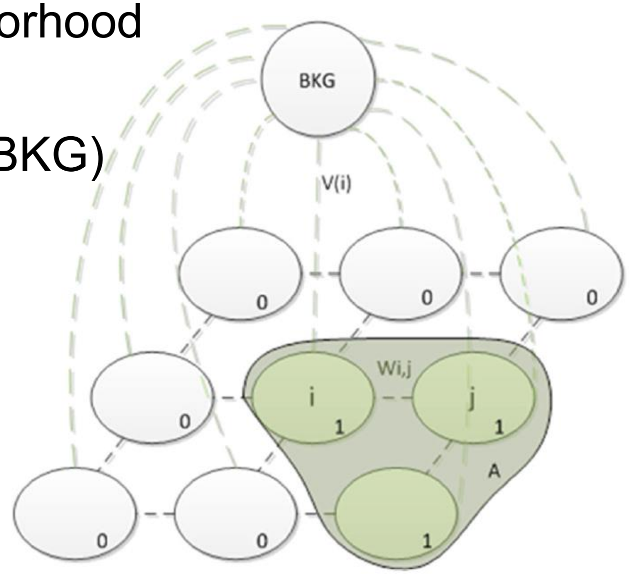


Figure 4. Visual comparison of previous approaches to our method and ground truth (GT).

Graph-based Saliency Detection

Quantum Cuts (QCut)-based saliency (extended version):

- Apply SLIC for defining super-pixels
- Represent each super-pixel using the mean LAB color value within it
- Define an undirected graph G(**V**,**E**), where the nodes **V** $\in \mathbb{R}^{DxN}$ of the graph are the N super-pixels of the image and the edges E connecting them are weighted using a weighting matrix **W** $\in \mathbb{R}^{NxN}$ expressing pair-wise similarities between graph nodes (we also use neighborhood information)
- Add an additional node representing the background (BKG)



Graph-based Saliency Detection

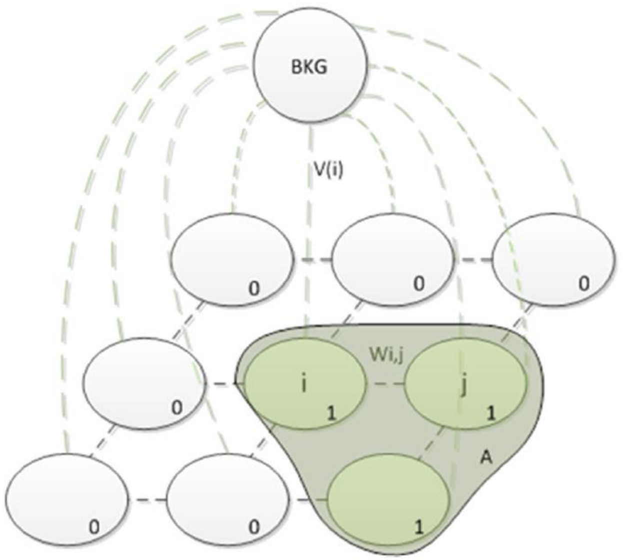
Quantum Cuts (QCut)-based saliency (extended version):

- We denote by A the foreground segment (the salient object(s))
- A needs to be in contrast with the rest of the graph. This means that we want

$$cut(A,\bar{A}) = \sum_{u \in A, v \in \bar{A}} w_{u,v}$$

to be minimized. We express the above as:

$$\mathrm{cut}\left(A,\bar{A}\right) = \mathrm{cut_U}\left(A,\bar{A}\right) + \mathrm{cut_B}\left(A,\bar{A}\right)$$
$$\mathrm{cut_B}\left(A,\bar{A}\right) = \sum_{i,j} w_{i,j}\left(y_i\left(1-y_j\right)\right)$$
$$\mathrm{cut_U}\left(A,\bar{A}\right) = \sum_{i} V(i) y_i$$



where y is a binary vector having values $y_i = 1$ if $\mathbf{v}_i$ is a super-pixel in the boundary of the image and $y_i = 0$, otherwise

Graph-based Saliency Detection

Quantum Cuts (QCut)-based saliency (extended version):

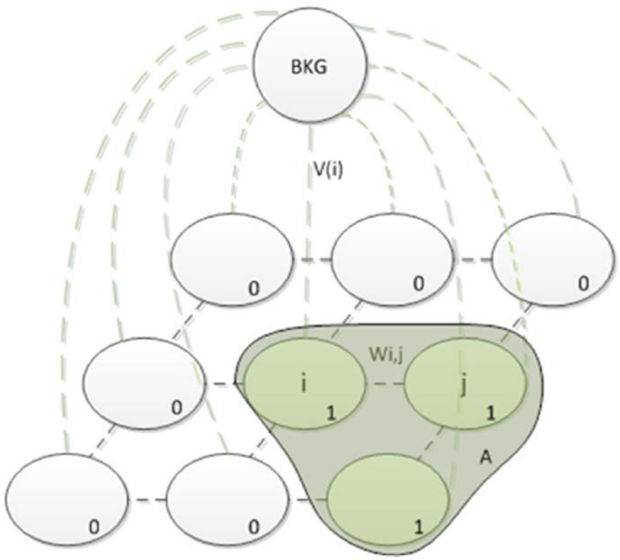
- We denote by A the foreground segment (the salient object(s))
- A needs to be in contrast with the rest of the graph. This means that we want

$$cut(A,\bar{A}) = \sum_{u \in A, v \in \bar{A}} w_{u,v}$$

to be minimized.

- A also needs to be (relatively) big in area
- Thus, we want to find A such that

$$\underset{A}{\operatorname{argmin}} \frac{\operatorname{cut}(A,\bar{A})}{\operatorname{area}(A)} \Rightarrow \underset{y}{\operatorname{argmin}} \frac{\sum_{i,j} w_{i,j}\left(y_j - y_i y_j\right) + \sum_i V(i) y_i}{\sum_i y_i}$$



Graph-based Saliency Detection

Quantum Cuts (QCut)-based saliency (extended version):

- We denote by A the foreground segment (the salient object(s))
- Thus, we want to find A such that

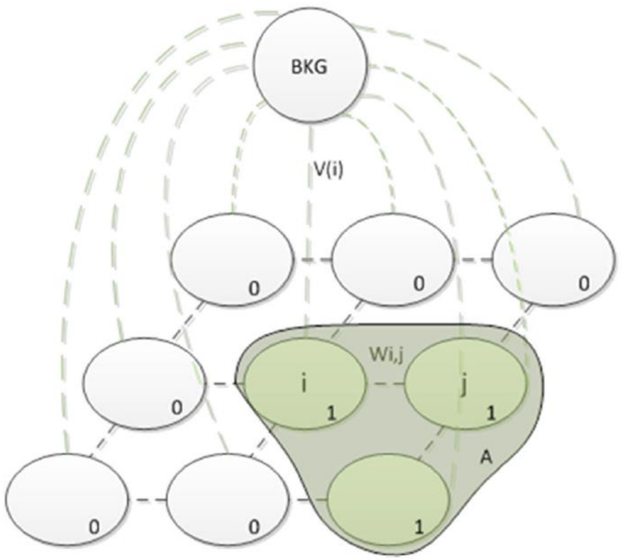
$$\underset{A}{\operatorname{argmin}} \frac{\operatorname{cut}(A,\bar{A})}{\operatorname{area}(A)} \Rightarrow \underset{y}{\operatorname{argmin}} \frac{\sum_{i,j} w_{i,j}\left(y_j - y_i y_j\right) + \sum_i V(i) y_i}{\sum_i y_i}$$

- We consider an auxiliary vector **z** satisfying:

$$z_i^2 = y_i, \text{ that is } z_i \in \{-1,0,1\}$$

and we add the constraint that the following quantity needs to be minimized (remember that $y_i = \{0,1\}$)

$$\sum_{i,j} w_{i,j}(z_i^2 z_j^2 - z_i z_j)$$



Graph-based Saliency Detection

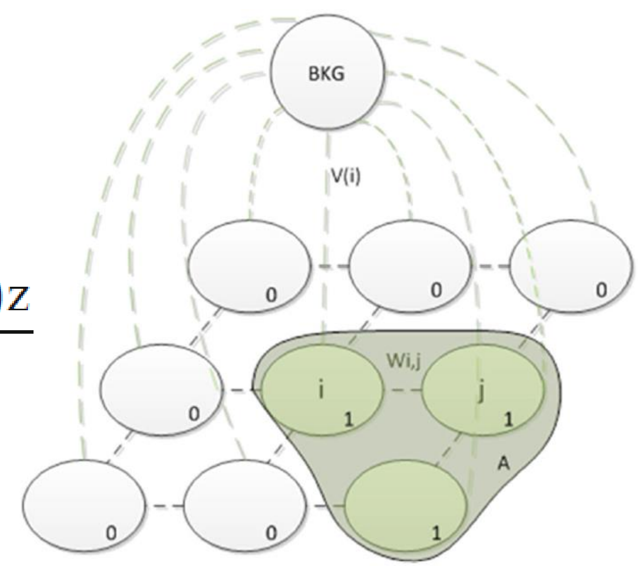
Quantum Cuts (QCut)-based saliency (extended version):

- We denote by A the foreground segment (the salient object(s))
- Thus, we want to find A such that

$$\underset{A}{\operatorname{argmin}} \frac{\operatorname{cut}\left(A, \bar{A}\right)}{\operatorname{area}(A)} \rightarrow \underset{y}{\operatorname{argmin}} \frac{\sum_{i,j} w_{i,j}\left(y_j - y_i y_j\right) + \sum_i V(i) y_i}{\sum_i y_i}$$

$$\underset{y}{\operatorname{argmin}} \frac{\sum_{i,j} w_{i,j}\left(z_i^2 z_j^2 - z_i z_j\right) + \sum_{i,j} w_{i,j}\left(z_j^2 - z_i^2 z_j^2\right) + \sum_i V(i) z_i^2}{\sum_i z_i^2}$$

$$= \underset{y}{\operatorname{argmin}} \frac{\sum_{i,j} w_{i,j}\left(z_j^2 - z_i z_j\right) + \sum_i V(i) z_i^2}{\sum_i z_i^2} = \underset{y}{\operatorname{argmin}} \frac{z^T (H_m) z}{z^T z}$$



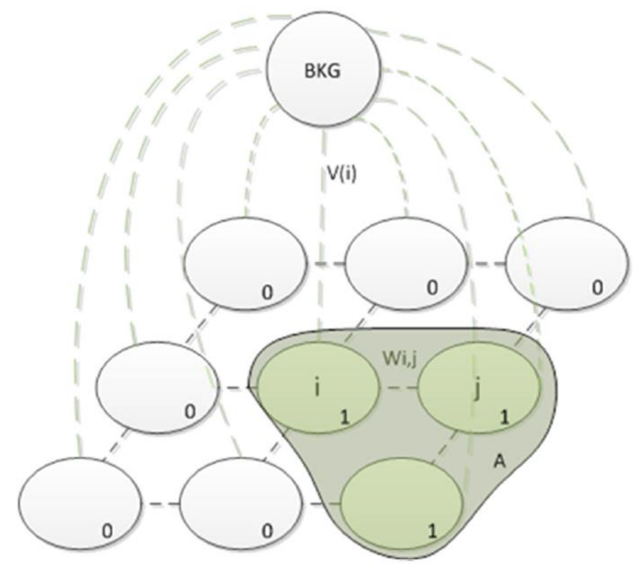
Graph-based Saliency Detection

Quantum Cuts (QCut)-based saliency (extended version):

- We denote by A the foreground segment (the salient object(s))
- Thus, we want to find A such that

$$\underset{y}{\operatorname{argmin}} \frac{z^T (H_m) z}{z^T z}$$

$$H_m(i,j) = \begin{cases} V(i) + \sum_{k \in N_i} w_{i,k} & i = j \\ -w_{i,j} & j \in N_i \\ 0 & e.w \end{cases}$$



Graph-based Saliency Detection

Quantum Cuts (QCut)-based saliency (extended version):

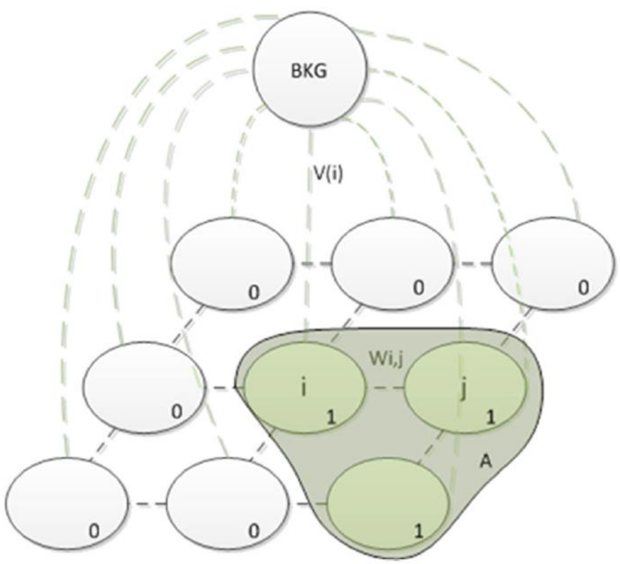
- We denote by A the foreground segment (the salient object(s))
- A is obtained by

$$\mathrm{H}_{\mathrm{m}} z_* = \mathrm{E}_m z_*$$

$$y_* = z_* \circ z_*$$

Eigen-analysis problem

Element-wise multiplication



Graph-based Saliency Detection

Quantum Cuts (QCut)-based saliency (extended version):

- Node connectivity using up to 5^{th} set of neighborhood:
  - Two super-pixels are m-neighbors if it is possible to reach one super-pixel from the other through m spatial connections.
  - the n-set neighbor of a super-pixel contains all super-pixels which are at least floor(2^{n-2}+1) and at most 2^{n-1} spatial connections
- Graph weights:

$$w_{j,i} = \left(\frac{1}{\varepsilon + d(LAB_i, LAB_j)^2}\right)\left(\frac{1}{|N_{i,C(i,j)}| * |N_j|}\right)^2$$

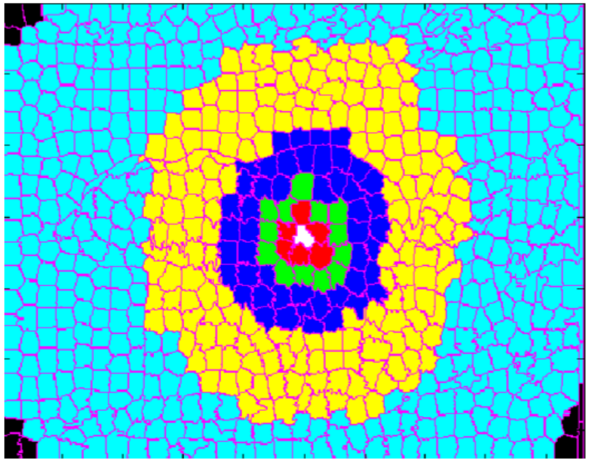
Euclidean distance

#connections of super-pixel i with C(i,j)

#neighbors of super-pixel j

C(i,j): the set of neighborhood level of nodes i and j

Fig. 4 Illustration of neighbourhood-sets. For the given superpixel (*white*), sets are shown as 1st (*red*), 2nd (*green*), 3rd (*blue*), 4th (yellow), 5th (*cyan*). Black superpixels do not correspond to any set



Graph-based Saliency Detection

Extension in multiple scales

Input: I: Image data in Lab Color Space, *scales*: number of different scales for SLIC.
Output: S: Saliency Map in gray-scale.
Steps:
1. Apply SLIC superpixel segmentation on I for scales
2. For $i = 1$: *scales* DO:
   - Assign high potential to V in Eq. (6) for image boundaries and 0 elsewhere.
   - Obtain affinities as in Eq. (13).
   - Construct $\boldsymbol{H_m}$ matrix, as in Eq. (6)
   - Compute the eigenvector of $\boldsymbol{H_m}$ with the smallest eigenvalue, i.e., the ground state wavefunction, $\boldsymbol{\psi}$.
   - Compute the soft labeling vector $y_i = \psi \circ \psi$,
3. Compute the average map of all scales and obtain the saliency map $S = \frac{1}{scales} \sum_{i=1}^{scales} y_i$

$$H_m(i,j) = \begin{cases} V(i) + \sum_{k \in N_i} w_{i,k} & i = j \\ -w_{i,j} & j \in N_i \\ 0 & e.w \end{cases}$$

$$w_{j,i} = \left(\frac{1}{\varepsilon + d(LAB_i, LAB_j)^2} \right) \left(\frac{1}{|N_{i,C(i,j)}| * |N_j|} \right)^2$$

Graph-based Saliency Detection

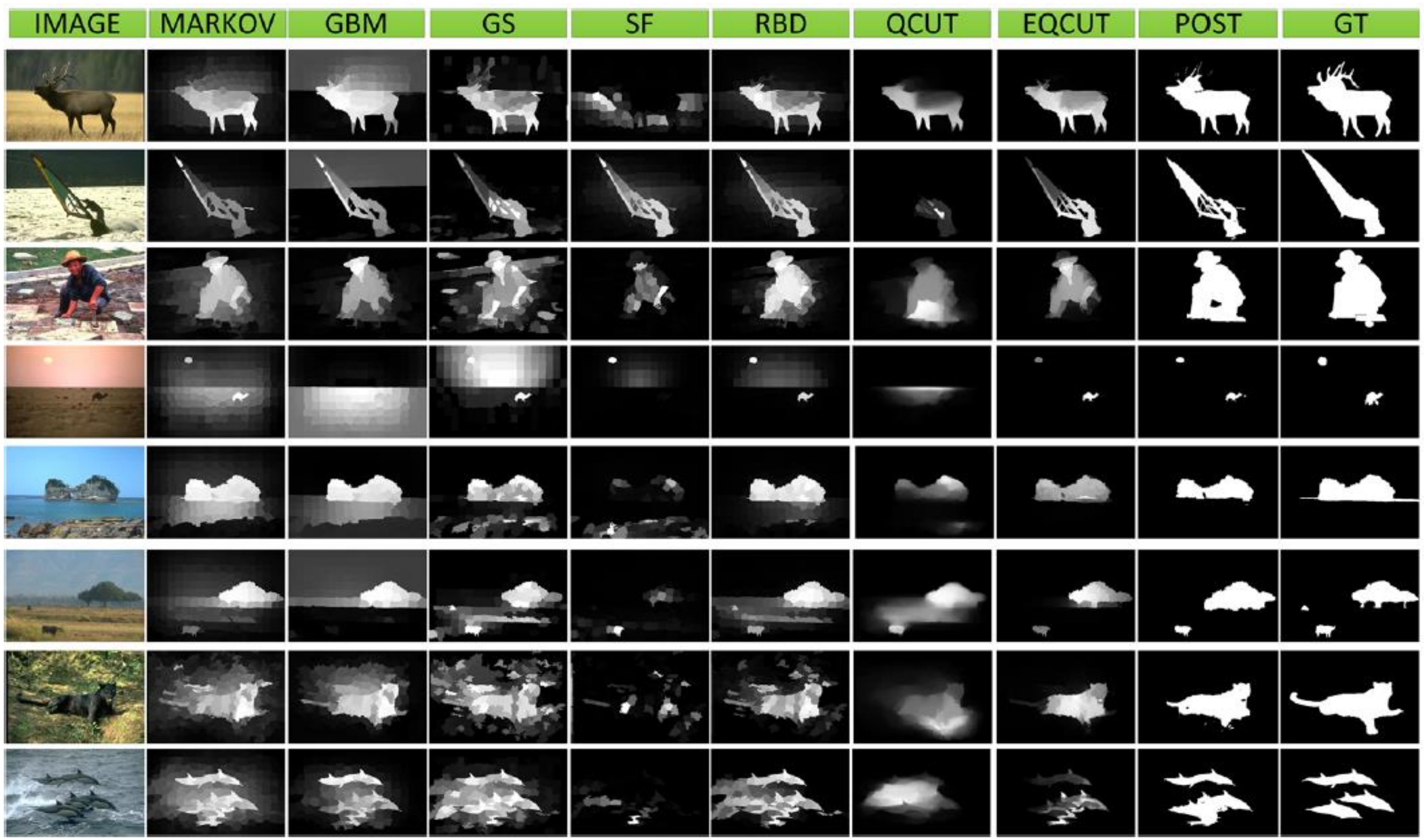


Fig. 6 Image, saliency maps extracted by MARKOV, GBM, GS, SF, RBD, QCUT, E-QCUT, segmentation on E-QCUT's result (*POST*), and ground truth (*GT*), from *left* to *right*

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$

$$\underset{\boldsymbol{P}(\mathrm{x})}{argmin} \left(\sum_i \left(\boldsymbol{P}(\mathrm{x}=x_i)\right)^2 v_i + \frac{1}{2} \sum_{i,j} \left(\boldsymbol{P}(\mathrm{x}=x_i) - \boldsymbol{P}\left(\mathrm{x}=x_j\right)\right)^2 w_{i,j} \right)$$
$$s.t. \quad \sum_{\boldsymbol{i}} \boldsymbol{P}(\mathrm{x}=x_i) = 1 \ .$$

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$

$$\underset{\boldsymbol{P}(\mathrm{x})}{\operatorname{argmin}} \left(\boxed{\sum_i \left(\boldsymbol{P}(\mathrm{x}=x_i)\right)^2 v_i} + \frac{1}{2} \sum_{i,j} \left(\boldsymbol{P}(\mathrm{x}=x_i) - \boldsymbol{P}(\mathrm{x}=x_j)\right)^2 w_{i,j} \right)$$
$$s.t. \quad \sum_i \boldsymbol{P}(\mathrm{x}=x_i) = 1 \ .$$

Suppression of a region x_i to be selected as salient, if there is prior information (encoded in $v_i \geq 0$) that it belongs to the background

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$

$$\underset{\boldsymbol{P}(\mathrm{x})}{argmin} \left(\sum_i (\boldsymbol{P}(\mathrm{x}=x_i))^2 v_i + \boxed{\frac{1}{2} \sum_{i,j} \left(\boldsymbol{P}(\mathrm{x}=x_i) - \boldsymbol{P}(\mathrm{x}=x_j)\right)^2 w_{i,j}} \right)$$
$$s.t. \quad \sum_{\boldsymbol{i}} \boldsymbol{P}(\mathrm{x}=x_i) = 1 \;.$$

Force two regions x_i and x_j to have similar probabilities to be selected, if their similarity $w_{i,j}$ (measured in a feature space) is high

Graph-based Saliency Detection

<u>Probabilistic Saliency Estimation:</u>

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$

$$\underset{\boldsymbol{P}(\mathrm{x})}{argmin} \left(\sum_i \left(\boldsymbol{P}(\mathrm{x}=x_i)\right)^2 v_i + \frac{1}{2}\sum_{i,j} \left(\boldsymbol{P}(\mathrm{x}=x_i) - \boldsymbol{P}(\mathrm{x}=x_j)\right)^2 w_{i,j} \right)$$

$$\boxed{s.t. \quad \sum_{\boldsymbol{i}} \boldsymbol{P}(\mathrm{x}=x_i) = 1}.$$

As a probability mass function, it should sum to 1

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$
- For symmetric similarity functions ($w_{i,j} = w_{j,i}$), we can write:

$$\underset{\boldsymbol{P}(\mathrm{x})}{argmin} \left(\sum_i (\boldsymbol{P}(\mathrm{x} = x_i))^2 v_i + \left(\sum_{i,j} \left((\boldsymbol{P}(\mathrm{x} = x_i))^2 - \boldsymbol{P}(\mathrm{x} = x_i) \boldsymbol{P}\left(\mathrm{x} = x_j\right) \right) w_{i,j} \right) \right)$$

$$s.t. \quad \sum_{\boldsymbol{i}} \boldsymbol{P}(\mathrm{x} = x_i) = 1$$

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$
- For symmetric similarity functions ($w_{i,j} = w_{j,i}$), we can write:

$$\underset{\boldsymbol{P}(\mathrm{x})}{argmin} \left(\sum_i (\boldsymbol{P}(\mathrm{x}=x_i))^2 v_i + \left(\sum_{i,j} \left((\boldsymbol{P}(\mathrm{x}=x_i))^2 - \boldsymbol{P}(\mathrm{x}=x_i)\boldsymbol{P}(\mathrm{x}=x_j) \right) w_{i,j} \right) \right)$$
$$s.t. \quad \sum_{\boldsymbol{i}} \boldsymbol{P}(\mathrm{x}=x_i) = 1$$

$$p_i = \boldsymbol{P}(\mathrm{x}=x_i) \Longrightarrow \boldsymbol{p}^* = \underset{\boldsymbol{p}}{argmin} \left(\boldsymbol{p}^T \boldsymbol{H} \boldsymbol{p} \right) \quad s.t. \ \boldsymbol{p}^T 1 = 1$$

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- It defines a competition between different super-pixels for selecting the most salient ones
- This is expressed under a probabilistic framework, which allows modeling the probability selecting region x_i as the most salient one using the probability mass function $P(x=x_i)$
- For symmetric similarity functions ($w_{i,j} = w_{j,i}$), we can write:

$$\underset{\boldsymbol{P}(\mathrm{x})}{argmin} \left(\sum_i (\boldsymbol{P}(\mathrm{x}=x_i))^2 v_i + \left(\sum_{i,j} \left((\boldsymbol{P}(\mathrm{x}=x_i))^2 - \boldsymbol{P}(\mathrm{x}=x_i)\boldsymbol{P}\left(\mathrm{x}=x_j\right) \right) w_{i,j} \right) \right)$$

$$s.t. \quad \sum_{\boldsymbol{i}} \boldsymbol{P}(\mathrm{x}=x_i) = 1$$

$$p_i = \boldsymbol{P}(\mathrm{x}=x_i) \quad \Longrightarrow \quad \boldsymbol{p}^* = \underset{\boldsymbol{p}}{argmin} \; \left(\boldsymbol{p}^T \boldsymbol{H} \boldsymbol{p}\right) \quad s.t. \; \boldsymbol{p}^T 1 = 1$$

$$\Downarrow$$

$$\boldsymbol{p}^* = \frac{1}{1^T \mathbf{H}^{-1} 1} \mathbf{H}^{-1} 1 = \mathrm{q}(\mathbf{H}) \; \mathbf{H}^{-1} 1$$

Graph-based Saliency Detection

Probabilistic Saliency Estimation:

- Uses the same graph used in E-Qcut
- It is a generalization of Diffusion-based methods, expressed using a probabilistic framework, where **A** = **H**
- It is a better approximation of the graph-cut problem $\mathrm{A}^* = \operatorname*{argmin}_{\mathrm{A}} \frac{\mathrm{cut}(\mathrm{A}, \bar{\mathrm{A}})}{\mathrm{area}(\mathrm{A})}$ compared to E-QCut:

PSE: $\boldsymbol{p}^* = \frac{1}{c}\mathbf{H}^{-1}1, \qquad c = 1^{\boldsymbol{T}}\mathbf{H}^{-1}1$

E-QCut: $\mathbf{p}^*_{\boldsymbol{Qcut}} = \boldsymbol{z}_* \circ \boldsymbol{z}_*, \qquad \mathbf{H}\boldsymbol{z}_* = \mathrm{E}_m\boldsymbol{z}_*$

Graph-based Saliency Detection

Fig. 2. Qualitative comparison of unsupervised methods. (From left to right: Image (IM), and the saliency maps obtained by DSR [10], MC [16], RBD [15], ST [41], GP [36], QCUT [20] and our PSE, and the ground truth (GT).

Aggregation of various saliency maps

Given a set of M saliency maps:

- We denote by $S_i(x)$ the saliency score of pixel x in the i-th map, i=1,…,M
- The saliency of pixel x can be obtained by fusing information in all M maps:

$$S(x) = P(s_x = 1|\mathbf{f}_x) \propto \frac{1}{Z} \sum\nolimits_{i=1}^{M} \zeta(S_i(x))$$

$$\zeta_1(z) = z; \qquad \zeta_2(z) = \exp(z); \qquad \zeta_3(z) = -\frac{1}{\log(z)}$$

- Also, a weighted fusion scheme can be obtained by:

$$P(s_x = 1|\mathbf{f}_x; \lambda) = \sigma\left(\sum_{i=1}^{M} \lambda_i S_i(x) + \lambda_{M+1}\right)$$

where λ_i, i=1,…,M+1 is the set of parameters to be optimized and σ(·) is the soft-max function: $\sigma(z) = 1/(1+\exp(-z))$

Neural network-based saliency detection

Fully Convolutional Networks:

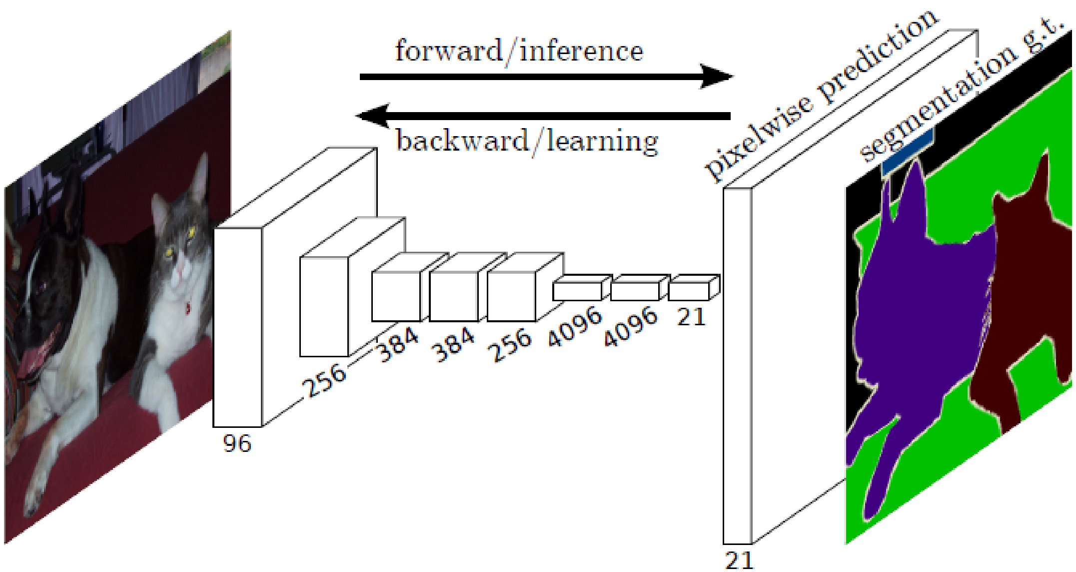


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Neural network-based saliency detection

Fully Convolutional Networks:

- While CNNs have been combined with MLP layers to solve classification problems, for semantic segmentation (at a pixel) level, we would like the output of the network to correspond to a map having the size (possibly resampled) of the input image.
- We can transform a standard CNN to a fully-convolutional network by transforming the MLP layers to convolutional layers with filter size of 1x1

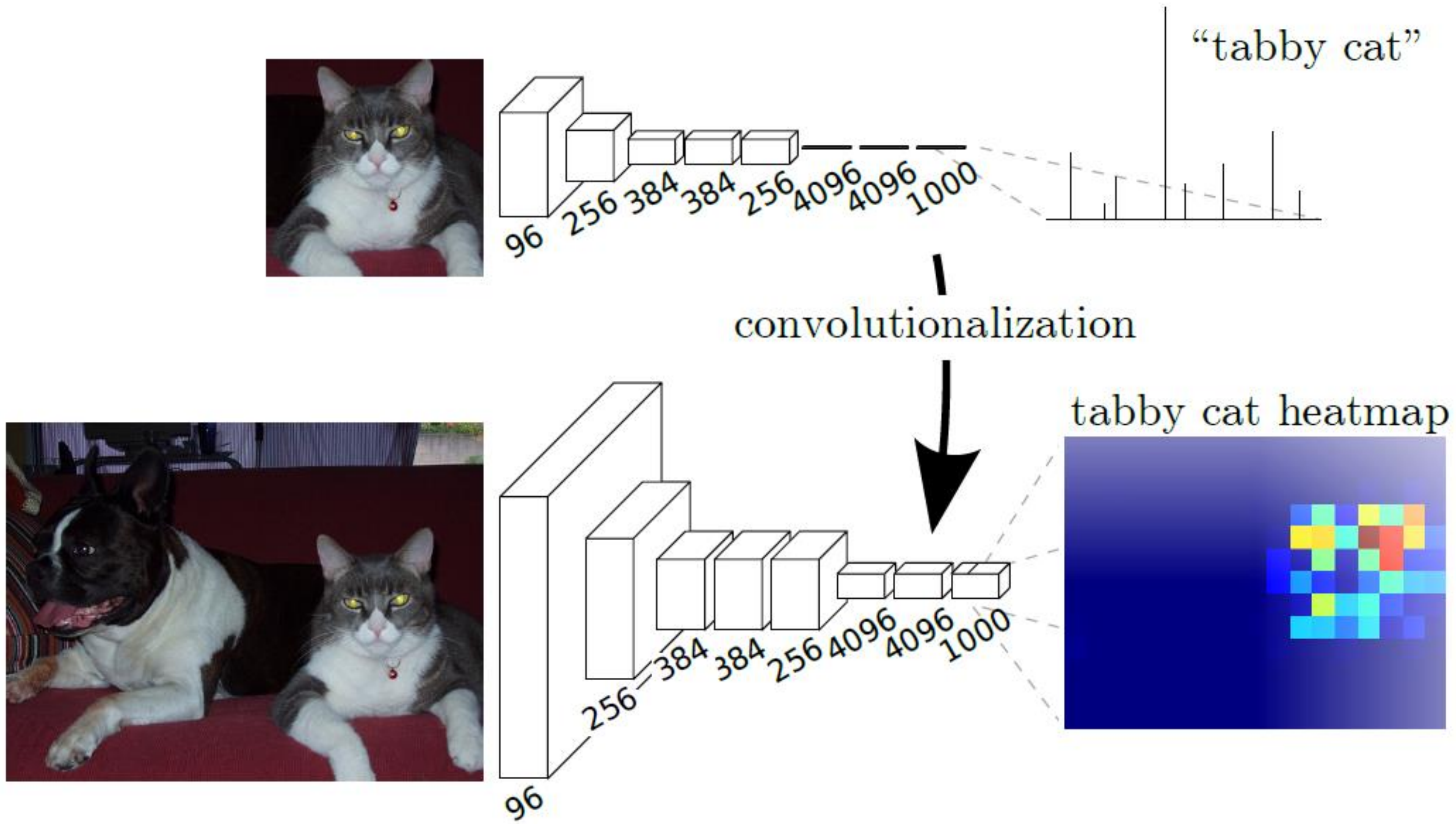
Neural network-based saliency detection

Fully Convolutional Networks:

- While CNNs have been combined with MLP layers to solve classification problems, for semantic segmentation (at a pixel) level, we would like the output of the network to correspond to a map having the size (possibly resampled) of the input image.
- We can transform a standard CNN to a fully-convolutional network by transforming the MLP layers to convolutional layers with filter size of 1x1
- In order to obtain output maps with the same resolution with the input image, up-sampling can be used:
  - bilinear interpolation
  - convolution with fractional input stride (1/f), which is known as de-convolution

Neural network-based saliency detection

Fully Convolutional Networks:



Neural network-based saliency detection

Fully Convolutional Networks:

- In order to combine information from different layers in the hierarchy (low-level features learned in the first layers of the CNN and high-level features learned in the later layers of the CNN), skip connections are used

Neural network-based saliency detection

Fully Convolutional Networks:

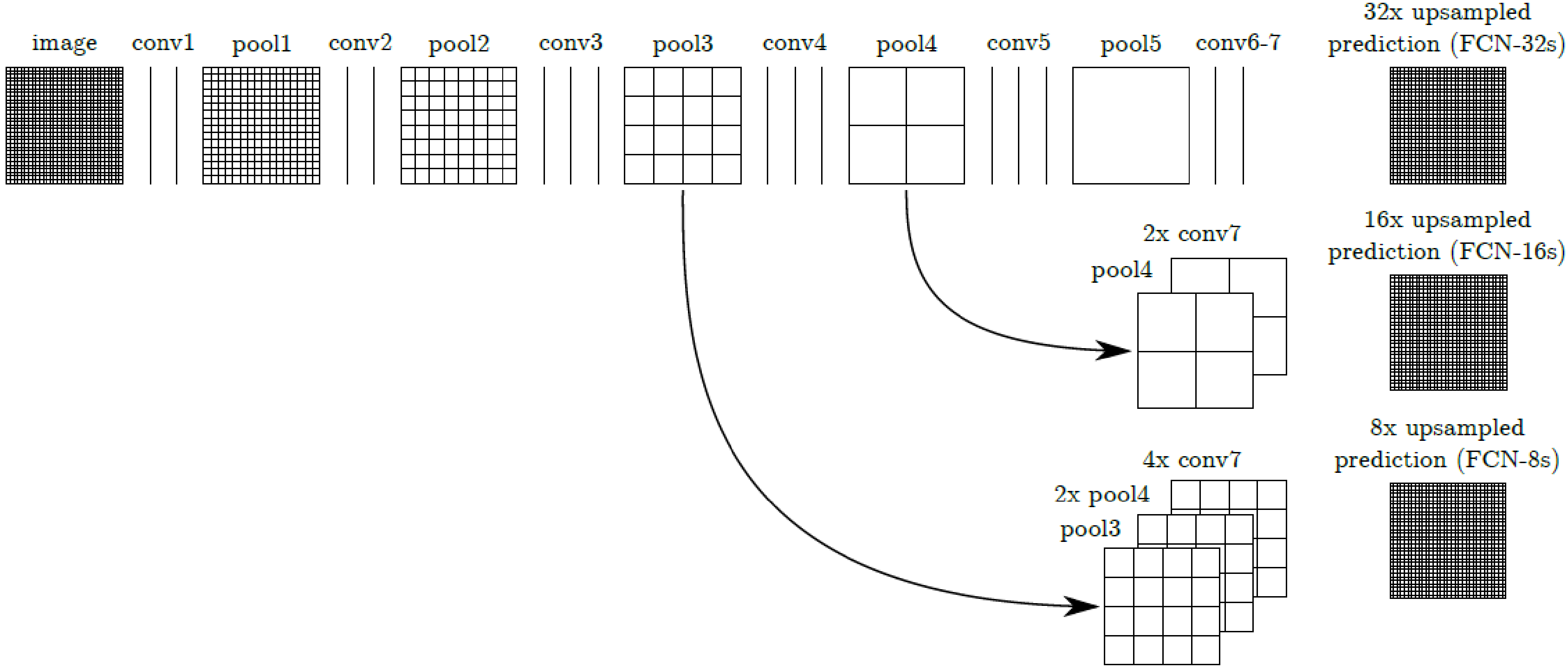


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

Neural network-based saliency detection

Fully Convolutional Networks:

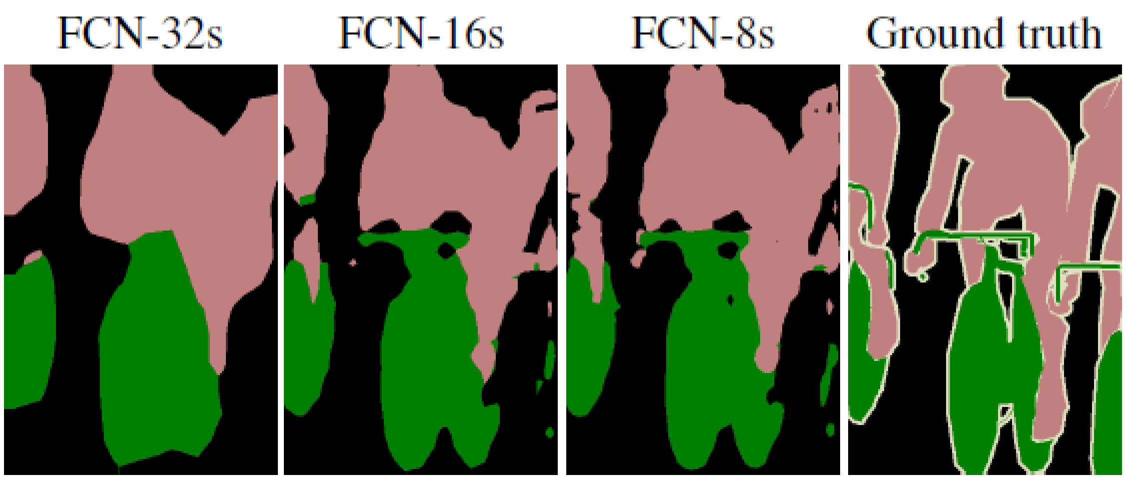


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Neural network-based saliency detection

Fully Convolutional Networks: many variants

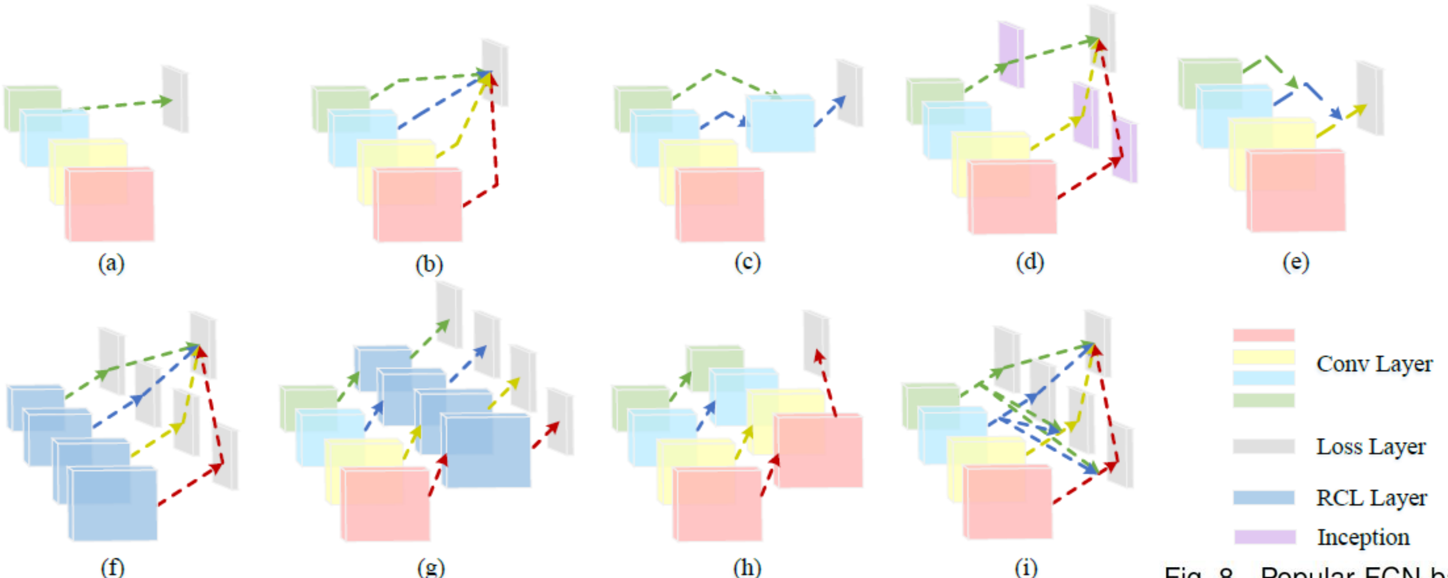


Fig. 8. Popular FCN-based architectures. One can see that apart from the classical architecture (a) more and more advanced architectures have been developed recently. Some of them (b,c,d,and e) exploit skip layers from different scales so as to learn multi-scale and multi-level features. Some of them (e, g, h, and i) adopt the encoder-decoder structure to better fuse high-level features with low-level ones. There are also some works (f, g, and i) introduce side supervision as done in [142] in order to capture more detailed multi-level information. See Table 9 for details on these architectures.

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

- Combination of local information (in the super-pixel level) and global image information
- Learning of optimized super-pixel representations based on supervised learning

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

Revisit of QCut: $\underset{y}{\operatorname{argmin}} \frac{z^T (H_m) z}{z^T z}$ $\quad H_m(i, j) = \begin{cases} V(i) + \sum_{k \in N_i} w_{i,k} & i = j \\ -w_{i,j} & j \in N_i \\ 0 & e.w \end{cases}$

The matrix H_m is positive semi-definite. This means that it can be expressed as $H_m = \Phi^T \Phi$, where $\Phi = [\varphi_1, \ldots, \varphi_N] \in \mathbb{R}^{DxN}$

Thus, each super-pixel i is represented by the corresponding φ_i.

Then, the similarity (affinity) between two super-pixels can be expressed using a kernel function of the form: $K(\varphi, \varphi') = \|\varphi(z)\|_H \, \|\varphi'(z')\|_H \, e^{-\frac{1}{2\sigma^2}\|\varphi(z) - \varphi'(z')\|_2^2}$

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

Then, the similarity (affinity) between two super-pixels can be expressed using a kernel function of the form:

$$K(\varphi, \varphi') = \|\varphi(z)\|_H \, \|\varphi'(z')\|_H \, e^{-\frac{1}{2\sigma^2}\|\varphi(z) - \varphi'(z')\|_2^2}$$

The above similarity can be approximated by learning l convolutional filters with parameters σ, η_l and $\boldsymbol{\mu}_l$, l=1,…,p leading to super-pixel representations:

$$\zeta : \boldsymbol{\varphi} \rightarrow \|\boldsymbol{\varphi}\|_2 \left[\sqrt{\eta_l} \, e^{-\frac{1}{\sigma^2}\|\boldsymbol{\varphi} - \boldsymbol{\mu}_l\|_2^2} \right]_{l=1}^{p}$$

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

Using the (learnable) super-pixel representation:

$$\zeta : \boldsymbol{\varphi} \rightarrow \|\boldsymbol{\varphi}\|_2 \left[\sqrt{\eta_l}\, e^{-\frac{1}{\sigma^2}\|\boldsymbol{\varphi}-\boldsymbol{\mu}_l\|_2^2} \right]_{l=1}^{p}$$

E-QCut is applied using the graph weight function:

$$w_{i,j} = \frac{1}{\varepsilon + 1 - \langle \tilde{\boldsymbol{\xi}}_i, \tilde{\boldsymbol{\xi}}_j \rangle}$$

The above-described process is differentiable with respect to the output.

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

Given an input image and a saliency mask (ground-truth) **g**:

- the image is over-segmented using SLIC
- Super-pixels are represented by using the vectors $\boldsymbol{\zeta}_i$, i=1,…,N
- Using the above representation, QCut produces the saliency map **y**
- The error between **g** and **y** is calculated using an error function
- Parameters are updated based on the (error) Back-Propagation $e = f(y, g)$

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

The above-described process is differentiable with respect to the output.

Derivatives of error w.r.t. the parameters:

$$\frac{\partial AvgErr}{\partial \boldsymbol{\mu}_l} = \sum_i \sum_j \frac{\partial AvgErr}{\partial w_{ij}} \frac{\partial w_{ij}}{\partial \boldsymbol{\mu}_l}, \quad \frac{\partial w_{ij}}{\partial \boldsymbol{\mu}_l} = 4 {w_{i,j}}^2 \frac{\eta_l}{\sigma^2} (\boldsymbol{c}_i + \boldsymbol{c}_j - 2 w_l) e^{-\frac{\|\boldsymbol{c}_i - \boldsymbol{\mu}_l\|_2^2 + \|\boldsymbol{c}_j - \boldsymbol{\mu}_l\|_2^2}{\sigma^2}}$$

$$\frac{\partial AvgErr}{\partial \eta_l} = \sum_i \sum_j \frac{\partial AvgErr}{\partial w_{ij}} \frac{\partial w_{ij}}{\partial \eta_l}, \quad \frac{\partial w_{ij}}{\partial \eta_l} = 2 {w_{i,j}}^2 e^{-\frac{\|\boldsymbol{c}_i - \boldsymbol{\mu}_l\|_2^2 + \|\boldsymbol{c}_j - \boldsymbol{\mu}_l\|_2^2}{\sigma^2}}$$

$$\frac{\partial AvgErr}{\partial \sigma} = \sum_i \sum_j \frac{\partial AvgErr}{\partial w_{ij}} \frac{\partial w_{ij}}{\partial \sigma}, \quad \frac{\partial w_{ij}}{\partial \sigma} = \frac{4}{\sigma^3} {w_{i,j}}^2 \, \eta_l (\|\boldsymbol{c}_i - \boldsymbol{\mu}_l\|_2^2 + \|\boldsymbol{c}_j - \boldsymbol{\mu}_l\|_2^2) e^{-\frac{\|\boldsymbol{c}_i - \boldsymbol{\mu}_l\|_2^2 + \|\boldsymbol{c}_j - \boldsymbol{\mu}_l\|_2^2}{\sigma^2}}$$

Neural network-based saliency detection

Saliency detection using Convolutional Kernel Networks and QCut:

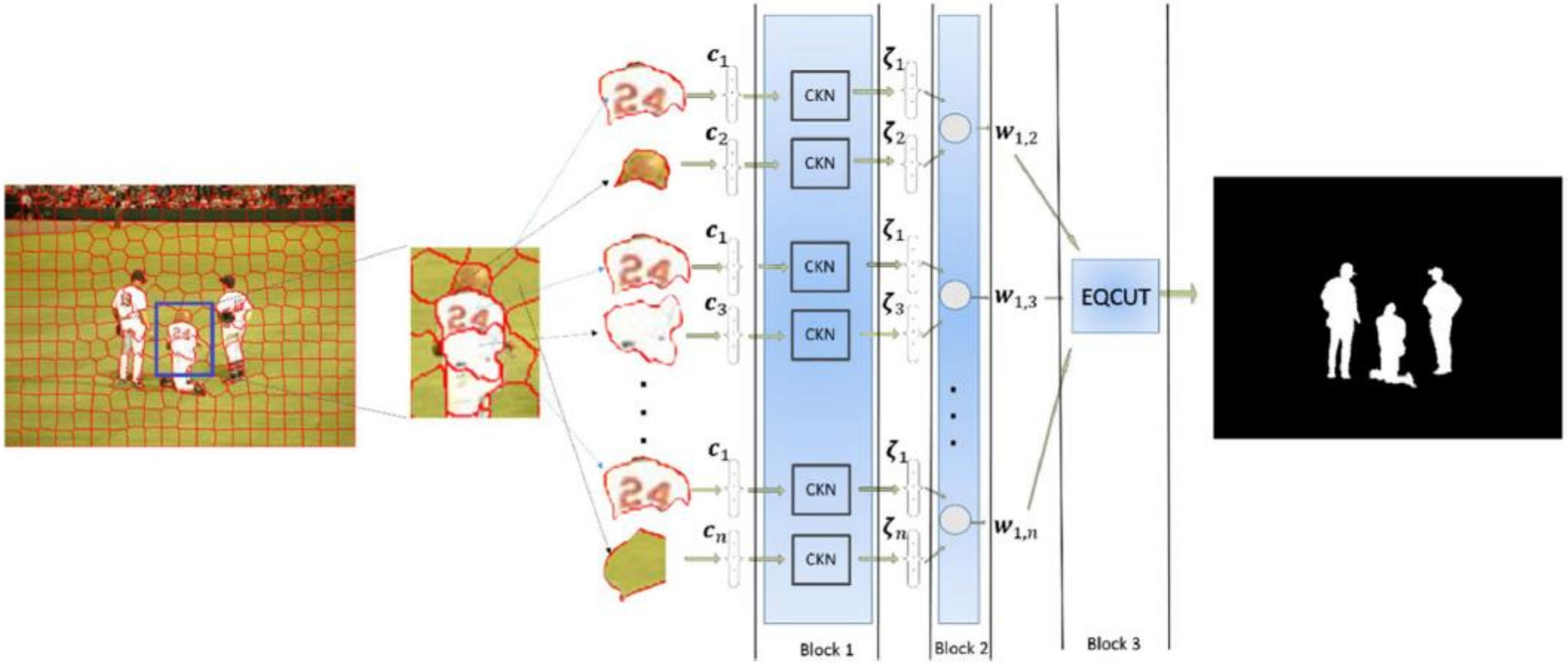


Fig. 2. The proposed salient object detection method: (1) Mean colors c of superpixel regions are fed to CKNs and the transformed features ζ are obtained, (2) Pairwise features are used to calculate affinities via Eqs. (9) and (3) The saliency map is calculated via EQCut applied on the affinity matrix $\mathbf{W}$.