# A Fully Convolutional Two-Stream Fusion Network for Interactive Image Segmentation

Yang Hu[a], Andrea Soltoggio[a], Russell Lock[a], Steve Carter[b]

[a] *Loughborough University, UK*
[b] *The ICE Agency, UK*

**Abstract**

In this paper, we propose a novel fully convolutional two-stream fusion network (FCTSFN) for interactive image segmentation. The proposed network includes two sub-networks: a two-stream late fusion network (TSLFN) that predicts the foreground at a reduced resolution, and a multi-scale refining network (MSRN) that refines the foreground at full resolution. The TSLFN includes two distinct deep streams followed by a fusion network. The intuition is that, since user interactions are more direct information on foreground/background than the image itself, the two-stream structure of the TSLFN reduces the number of layers between the pure user interaction features and the network output, allowing the user interactions to have a more direct impact on the segmentation result. The MSRN fuses the features from different layers of TSLFN with different scales, in order to seek the local to global information on the foreground to refine the segmentation result at full resolution. We conduct comprehensive experiments on four benchmark datasets. The results show that the proposed network achieves competitive performance compared to current state-of-the-art interactive image segmentation methods[1].

*Keywords:* Interactive image segmentation, Fully convolutional network, Two-stream network

## 1. Introduction

Binary image segmentation aims to separate an image into an object of interest (foreground) and the other parts (background). It has a wide range of applications, *e.g.*, medical image analysis, image editing, object retrieval, *etc.* However, since the object of interest varies highly in different contexts, most fully automatic methods are tailored and optimized to seek the particular object of interest in a certain application. It is difficult to develop a fully automatic method which is guaranteed to work in general applications.

To improve the flexibility and generality of image segmentation methods, many algorithms adopt interactive frameworks. These algorithms allow users to interact with a system to specify the object of interest by labeling some foreground/background pixels. Most traditional algorithms of interactive image segmentation [1, 2, 3, 4, 5, 6, 7] rely on low-level features to estimate the foreground/background distributions from user-labeled pixels, to predict the category of unlabeled pixels. A problem relating to these methods is that low level features may not be effective to distinguish between foreground and background in many situations, *e.g.*, the foreground and the background have similar color and texture; or the foreground includes several parts with very different appearance. Consequently, low-level feature-based algorithms may require a large number of user interactions to obtain reliable segmentation, increasing the burden on the end user.

Recently, deep features produced by deep neural networks (DNNs) have shown their power in many computer vision tasks including image classification [8, 9, 10] and semantic segmentation [11, 12, 13, 14]. Thus, several researchers [15, 16, 17, 18, 19, 20, 21] have used DNNs to extract deep features with higher-level understanding for image and user interactions to improve interactive image segmentation. Most of these DNN-based methods can be viewed as an early fusion of features using DNN. They concatenate features from image and user interaction as the input to DNN; generally, a DNN is used to combine the concatenated features to predict the foreground and background. However, such early-fusion schemes may not fully exploit the information in user interactions to predict foreground/background. Specifically, considering that state-of-the-art DNNs usually consist of a large number of layers, an early-fusion of user interactions with image features may weaken the influence of user interactions on the final prediction results.

In contrast to existing networks performing early fusion, we argue that better performance can be achieved with a late-fusion structure that uses two individual deep streams to learn and extract deep features from the image

*Email addresses:* y.hu@lboro.ac.uk (Yang Hu), a.soltoggio@lboro.ac.uk (Andrea Soltoggio), r.lock@lboro.ac.uk (Russell Lock), stevec@theiceagency.co.uk (Steve Carter)

[1] The codes will be available at https://github.com/cyh4/FCTSFN.

and the user interactions individually, then fusing the features from the two streams. Our intuition is that such a late fusion structure allows the user interactions to have a more direct impact on the prediction result, as it has a smaller number of layers between pure user interaction features and the prediction results. We expect this will lead to improved performance, as user interactions are more direct information on the location of foreground/background than the image itself. At the same time, deep features are still produced from the two individual deep streams, so the whole network still preserves the representative advantage of deep features. This allows the network to accurately understand image content and predict the object of interest.

In this paper, we propose a novel fully convolutional two-stream fusion network (FCTSFN) for interactive image segmentation. As shown in Fig. 1, the proposed network starts with two-stream late fusion network (TSLFN). The TSLFN extracts deep features from the image and the user interaction individually using two separate streams, and it applies a fusion net to fuse the features from the two streams to predict foreground and background. Since this two-stream late fusion structure reduces the number of layers between pure user interaction features and the network output, we expect it is able to improve the impact of user interactions on the prediction results to achieve better segmentation performance. Furthermore, to handle the loss of resolution in the TSLFN, we use a multi-scale refining network (MSRN) to refine the result of TSLFN at full resolution. The MSRN fuses the features from different layers of the TSLFN with different scales. It is expected that the fusion result includes the local to global structural information on the object specified by user interactions, and hence the MSRN can utilize the fusion result to refine the ouput of TSLFN at full resolution. The contributions of this paper are as follows:

- We propose a novel fully convolutional two-stream fusion network (FCTSFN) for interactive image segmentation.
- In FCTSFN, we propose a two-stream late fusion network (TSLFN) that aims to improve the impact of user interactions on the prediction results to achieve better segmentation performance.
- In FCTSFN, we propose a multi-scale refining network (MSRN) that fuses the information at different scales to refine the output of the TSLFN.

The rest of the paper is organized as follows. In section 2, we review related works. In section 3, we detail the proposed FCTSFN for interactive image segmentation. In section 4, we report the experimental results of the analysis and of the comparisons. Section 5 concludes the paper.

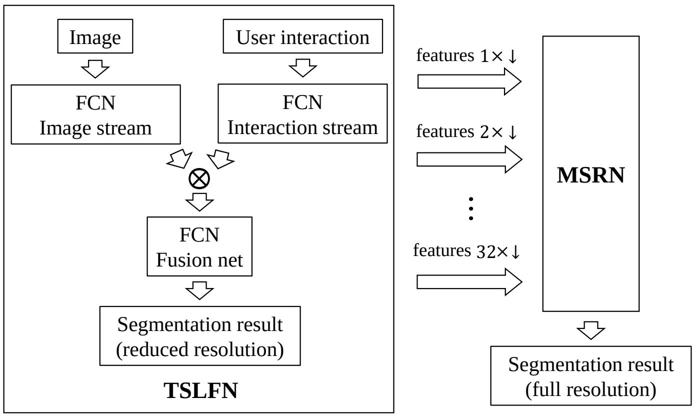


Figure 1: Flowchart of the proposed network architecture. ⊗ is the concatenation operator; $n\times\downarrow$ means $n$-time downsampling

## 2. Related works

A large number of methods have been proposed for interactive image segmentation. Boykov and Jolly [1] propose a graph cut-based method. This method represents the image as a graph where pixels are considered as graph nodes and neighbouring nodes are connected by edges. With this graph structure, interactive image segmentation is formulated as an energy minimization problem which can be solved by graph cuts [22, 23]. Following [1], Rother *et al.* [2] propose GrabCut which applies graph cuts iteratively. With an initial bounding box provided by the user, GrabCut iterates between foreground/background distribution estimation and graph cut segmentation to progressively refine foreground and background. With a similar graph representation to that in [1], Bai and Sapiro [3] determine foreground and background using the geodesic distance between unlabeled pixels and user-labelled foreground/background pixels. To take the advantage of both graph cut and geodesic distance, Price *et al.* [4] propose geodesic graph cut which incorporates geodesic distance into a graph cut-based framework. Also, Gulshan *et al.* [5] improve the graph cut method [1] by applying geodesic distance with star convexity as a shape constraint for the segmentation result. In other representative research, Vezhnevets and Konouchine [6] propose GrowCut which iteratively updates labels of pixels based on cellular automaton [24]. Grady [7] proposes a random walk method. This method calculates the probability that a random walker firstly reaches user-labelled pixels when starting from each unlabeled pixel; the pixel labels are then assigned based on the user-labeled pixel with the highest probability. All the above methods utilize low level features (color, texture, *etc.*) to model the foreground and background distributions. Therefore, their performance is restricted by the suitability of low-level features to distinguish between the foreground and background. As a result, for complex scenes where low-level features are less descriptive of the foreground and background differences, these methods may need users to label a large number of pixels to achieve good segmentation result. This increases the load of users.

Recently, deep neural networks (DNNs) have shown superior performance in distinguishing different objects in images [11, 12, 13, 14]. Also, the deep features learned from DNNs are proven to be highly transferable to other problems [25, 26]. Hence, several researchers have focused on applying DNN to gain features with higher-level understanding for image and user interactions to improve interactive image segmentation. Xu *et al.* [15] use two Euclidean distance maps to represent positive and negative clicks of the user. They form image-user interaction pairs by concatenating the two distance maps with RGB channels of the image. A fully convolutional network (FCN) is trained to predict the foreground/background from image-user interaction pairs. With similar image-user interaction pairs as input to the network, Boroujerdi *et al.* [17] use a lyncean fully convolutional network to predict foreground/background. This network replaces the last two convolutional layers in the FCN in [15] with three convolutional layers with gradually decreased kernel size to better capture the geometry of objects. Wang *et al.* [16] transform user interactions into two geodesic distance maps. They construct image-user interaction pairs similarly to [15] but augment it with an initial segmentation proposal produced by an additional DNN. They predict foreground/background using a resolution-preserving network. Xu *et al.* [18] propose Deep GrabCut. This method can seek the object boundary from a bounding box provided by the user. It transfers the bounding box into a distance map. An encoder-decoder network is used to predict foreground/background from the concatenated image and distance map. Maninis *et al.* [19] seek the foreground from extreme points. They encode extreme points into 2D Gaussians which are concatenated with the input image; a residue network [10] and a pyramid scene parsing model [27] are used to predict the foreground. Li *et al.* [28] use a segmentation network to generate various potential segmentations from image and user interactions; then a selection network is applied to select the output from the potential segmentations. Mahadevan *et al.* [21] propose an iterative training algorithm. Instead of training with fixed user clicks, this algorithm adds clicks progressively based on the error of the network predictions. This algorithm leads to improved performance, since it is more closely aligned with the patterns of real users. Essentially, all these networks [15, 17, 16, 18, 19, 21, 28] adopt early fusion structures. They combine the image and the user interaction features from the first layer of DNN. Differently from them, the proposed FCTSFN extracts deep image and user interactions features individually and then fuse them.

The work most similar to the proposed network is that of Liew *et al.* [20]. This is also a two-branch network: it includes a global branch producing coarse global predictions and a local branch utilizing multi-scale spatial pyramid features to make refined local predictions; the final prediction is the combined results from the two branches. However, the proposed network differs from the network in [20] in three important aspects. First, Liew *et al.* concatenate the image and interaction maps as the input of the network; the proposed network uses two individual streams to extract features from the image and interaction maps, to allow user interactions to have a more direct impact on the segmentation results. Second, Liew *et al.* produce multi-scale features using a spatial pyramid pooling on the features at an end layer of the network; the proposed network utilizes and fuses the features from different layers of the network, to incorporate both low-level information like color and edges and higher-level object information into the foreground prediction. Third, Liew *et al.* use multi-scale features to refine local segmentations which are then combined with global segmentation; the proposed network uses the multi-scale features in a direct global prediction refinement structure to make the prediction at full-resolution.

## 3. The proposed network

In this section, we present the proposed fully convolutional two-stream fusion network (FCTSFN) for interactive image segmentation. Firstly, we describe the architecture of the two-stream late fusion network (TSLFN) in the overall FCTSFN architecture. Then, we present the structure of the multi-scale refining network (MSRN) in the FCTSFN. Next, we demonstrate the network training process. Finally, we describe the data processing for the whole FCTSFN, including the method to generate user interaction maps from user interactions as input to the network and the method to produce the foreground mask based on the output of the network.

### *3.1. Two-stream late fusion network (TSLFN)*

Fig. 2(a) presents the structure of the TSLFN in the proposed FCTSFN. The input of TSLFN has two parts: one is an image and the other one is the concatenated positive and negative interaction maps generated respectively from positive and negative user interactions (see section 3.4). The network outputs a foreground probability map at a reduced resolution indicating the likelihood that a pixel is foreground. This network uses the VGG16 network [9] as the base network. It includes three parts: an image stream, an interaction stream, and a fusion net. The network in either image or interaction stream consists of the first 10 convolutional layers of the VGG16 network with rectified linear unit (ReLU) activation. After several convolutional layers, a max-pooling layer is applied with a kernel size of $2 \times 2$ and a stride of 2. The intent of the two streams is to learn deep features for images and interaction maps individually.

At the end of the image and the interaction streams, the feature maps from both streams are concatenated.

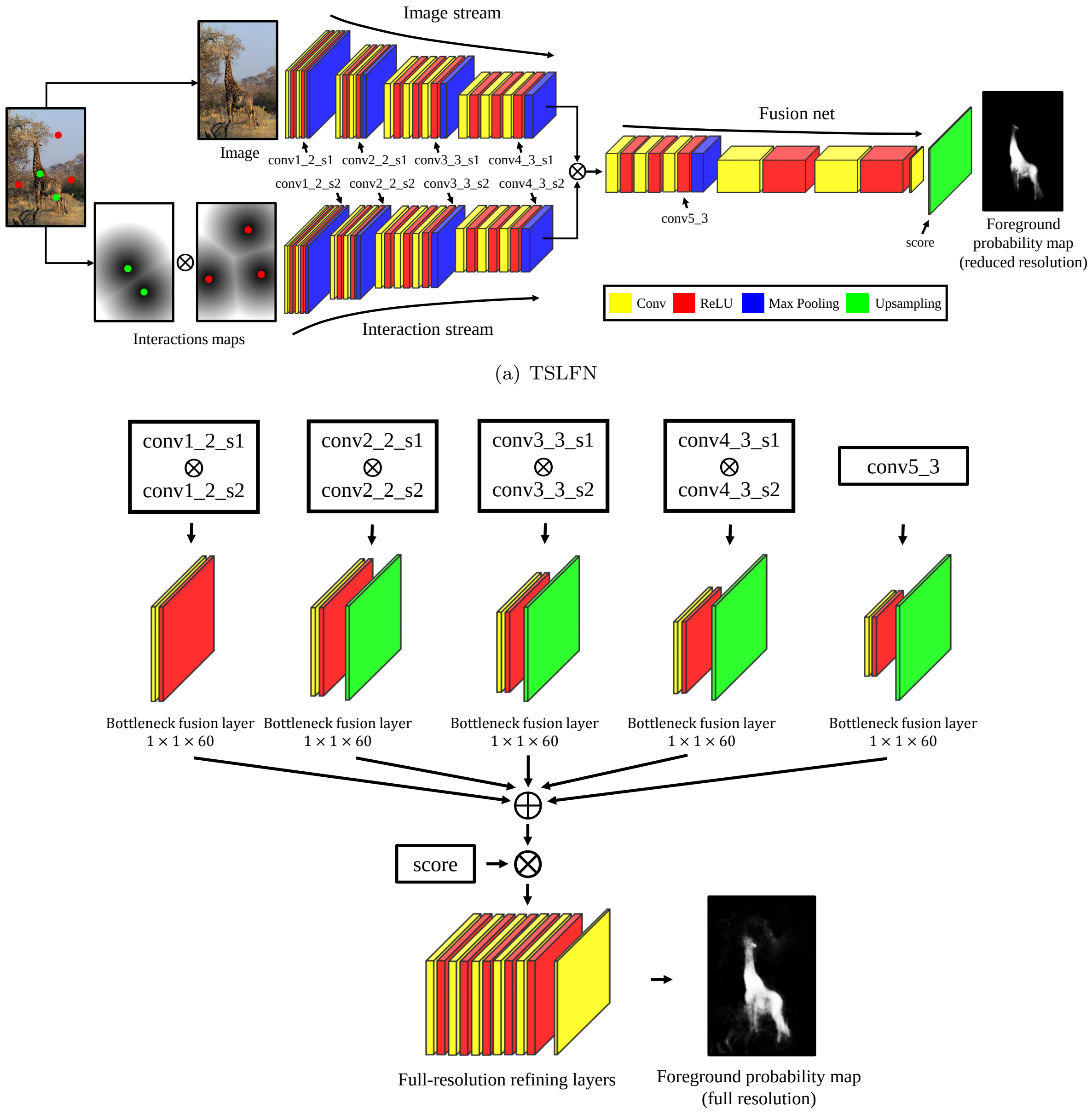


Image stream
Image
conv1_2_s1
conv2_2_s1
conv3_3_s1
conv4_3_s1
conv1_2_s2
conv2_2_s2
conv3_3_s2
conv4_3_s2
Fusion net
conv5_3
score
Foreground probability map (reduced resolution)
Interactions maps
Interaction stream
Conv
ReLU
Max Pooling
Upsampling
conv1_2_s1 ⊗ conv1_2_s2
conv2_2_s1 ⊗ conv2_2_s2
conv3_3_s1 ⊗ conv3_3_s2
conv4_3_s1 ⊗ conv4_3_s2
conv5_3
Bottleneck fusion layer 1 × 1 × 60
Bottleneck fusion layer 1 × 1 × 60
Bottleneck fusion layer 1 × 1 × 60
Bottleneck fusion layer 1 × 1 × 60
Bottleneck fusion layer 1 × 1 × 60
score
Full-resolution refining layers
Foreground probability map (full resolution)

(a) TSLFN

(b) MSRN

Figure 2: Architectures of the proposed fully convolutional two-stream fusion network (FCTSFN). It includes a two-stream late fusion network (TSLFN) and a multi-scale refining network (MSRN). ⊕ is element-wise sum operation. The sizes of convolutional layers are in the format of filter height × filter width × number of filters

The fusion net is then applied to learn to combine the concatenated feature maps to predict foreground/background. The fusion net consists of 6 convolutional layers: the first 3 of them are from the last 3 convolutional layers of the VGG16 network (corresponding to `conv5_1`, `conv5_2`, `conv5_3` in the VGG16); the last 3 of them are transferred from the fully connected layers in the VGG16 network using the method in [11]. Since the output of the whole network is downsampled by a factor of 32 with respect to the input image, we use an upsampling layer to upscale the output to the original resolution. Similarly to [11], the upsampling layer is a deconvolution layer with the filters set to bilinear interpolation kernels.

Note that variations of this TSLFN structure can be devised. One can make different assignments of the layers of the VGG16 network between the image/interaction stream and the fusion net to create variations of TSLFN with different depth in the two streams and the fusion net. This is essentially a trade-off between the impact of user interactions and the prediction capacity of the network. If we use fewer layers in the fusion net, the location information in user interactions may have higher impact on the prediction results, since it reduces the number of layers between pure user interaction features and the prediction result. However, this may harm the prediction capacity of the network, since the fusion net is too shallow and it may not be able to learn an effective projection from image/user interaction features to the foreground. On the other hand, a deeper fusion net may have sufficient capacity to learn a projection from image/user interaction features to the foreground, but it may weaken the influence of the user interactions on the prediction result due to the increasing number of layers between the pure user interaction features and the prediction result. Experimentally, we find that the structure shown in Fig. 2(a) provides top performance compared to its other variations (see section 4.2). We assume that this is because it achieves the best trade-off between the impact of user interactions and the prediction capacity, given our base network.

Also, we note that another possible way to fuse the image and user interaction features is the layer-by-layer fusion proposed by Hazirbas *et al.* [29]. This method uses two individual streams and fuses the features from the two streams multiple times at different layers. We have conducted experiments with this fusion architecture in early stage of our experiments (*i.e.* we perform similar layer-by-layer fusion between the image and user interaction streams with our base network). We found that it led to a performance drop for the task of interactive image segmentation. Considering that this architecture is originally designed to handle RGB-depth (RGBD) data, we think it is the differences in the characteristics of data that lead to the performance drop. For the RGBD data, the depth data includes accurate object boundary information, hence fusing it layer-by-layer with images data enhances the object boundary information, as discussed in [29]. However, for our task of interactive segmentation, the interaction maps do not include such accurate object boundary information; as a result, it is possible that fusing the features in the interaction stream into the image stream layer-by-layer brings difficulties for the network to learn information about objects.

### *3.2. Multi-scale refining network (MSRN)*

The MSRN in the proposed FCTSFN aims to fuse the information at different scales in the TSLFN in order to gain a better understanding of the location of the foreground and refine the predicted foreground at full resolution. Fig. 2(b) presents the architecture of the MSRN. The MSRN uses the features in six different scales from the TSLFN: the concatenated feature maps before each pooling layer of the image and the interaction streams (see `conv1_2_s1`, `conv1_2_s2`, `conv2_2_s1`, `conv2_2_s2`, `conv3_3_s1`, `conv3_3_s2`, `conv4_3_s1`, `conv4_3_s2` in Fig. 2); the feature maps before the pooling layer in the fusion net (`conv5_3` in Fig. 2); the upscaled prediction scores (`score` in Fig. 2). The features at each scale, except the predictions scores, are passed through a convolutional layer with a size of $1 \times 1 \times 60$ (filter height × filter width × number of filters), and the feature maps with downsampling are upscaled to the original resolution (the second row in Fig. 2(b)). We call these layers "bottleneck fusion layers" due to their two-fold effects. First, they fuse the feature maps from the image and the interaction streams to seek the information of the foreground on a specific scale. Second, they act as bottleneck layers to reduce the dimension of feature maps to keep the computational cost feasible.

After the bottleneck fusion layers, the feature maps from different scales are fused by an element-wise sum operation. Then, the fused features are concatenated with the prediction scores from the TSLFN. Finally, the concatenated features are passed through the full-resolution refining layers to predict the refined foreground (the last row in Fig. 2(b)). The full-resolution refining layers consist of a stack of six convolutional layers. The size of these convolutional layers are $7 \times 7 \times 64$, $5 \times 5 \times 64$, $3 \times 3 \times 64$, $3 \times 3 \times 64$, $3 \times 3 \times 64$, $1 \times 1 \times 2$. We use filters with large to small size to capture information from coarse to fine regions. The last convolutional layer works as a classifier.

### *3.3. Network training*

We train the FCTSFN in two stages. In the first stage, we remove the MSRN and fine-tune the TSLFN from the pre-trained VGG16 base network. In the second stage, we fix the parameters in the TSLFN and train the MSRN from scratch.

**Fine-tuning TSLFN**. We fine-tune the TSLFN (Fig. 2(a)) from the VGG16 network pre-trained on the ImageNet dataset [9, 30]. We use pixel-wise softmax loss. We adopt the "heavy" learning scheme in [11] using a batch size of 1 and a momentum of 0.99, due to the reported effectiveness of this method in fine-tuning FCN for image segmentation tasks [11]. At this stage, due to the differences in shapes, the pre-trained weights are not directly applicable to the following layers: the first convolutional layer in the interaction stream, the first convolutional layer in the fusion net, and the last convolutional layer in the fusion net. For the first convolutional layer in the interaction stream with a two-channel input, we use the mean of the filters in the first convolutional layer of the pre-trained VGG16 network to initialize it. The first convolutional layer in the fusion net has a doubled number of channels compared to the corresponding layer in VGG16 (`conv5_1`), due to the concatenation of feature maps from the two streams. To initialize this layer, we divide the channels of this layer into two halves and copy the pre-trained weights in `conv5_1` layer of VGG16 to each half. For the last convolutional layer in the fusion net, we initialize it with all zeros. Furthermore, we employ similar methods in [11] to fine-tune a stride-16 network and a stride-8 network for TSLFN to incorporate the features with finer scales to predict the foreground. We use the stride-8 network of TSLFN as the final form of TSLFN to make predictions with MSRN.

**Training MSRN from scratch**. With TSLFN parameters trained and fixed, we then train MSRN from scratch. We find that, at this stage, class-imbalance has a significant impact on the training performance. Specifically, the foreground usually occupies a relatively smaller region than the background in the task of interactive image segmentation. This leads to far more background pixels than the foreground pixels in our training data (see section. 4.1). Consequently, the learned network is easily biased to the background, resulting predictions with all pixels being background without foreground. To account for this problem, we crop the image centered at the foreground, if the area of the bounding box of the foreground occupies less than 35% of the image area. To avoid overfitting, we use three further strategies in the training. (1) Data augmentation: before a forward-backward pass, the training images used in this pass have a 50% probability to receive a random rotation, and a 50% probability to receive a random translation. (2) Dropout: each convolutional layer belonging to the full-resolution refining layers in the MSRN (see Fig. 2(b)) are followed by a dropout layer with a dropout ratio of 0.5, except the last layer for classification. (3) Early stopping: we record the validation accuracy on the validation data every 1000 iterations, and we terminate the training when we observe no improvement on accuracy in several consecutive validations. The other settings for the training of MSRN are as follows. We initialize all the convolutional layers randomly using the method in [31]. We resize all the training images to a resolution of 240 × 320 (height×width), and we train with a batch size of 3. We set the initial learning rate to 1e−8, weight decay to 0.0005, and momentum to 0.99.

### *3.4. Data processing*

The data processing in this paper includes two parts: the generation of user interaction maps and the postprocessing of the network output. We employ the method in [15] to generate interaction maps from user clicks as input to the network. Given an image and user clicks, the sets of positive and negative clicks are transferred into a positive and a negative interaction map, respectively. Either interaction map has the same height and width as the input image. Let $\mathcal{S}$ be a set of either positive or negative clicks. Let $s_{ij} \in \mathcal{S}$ be a click in $\mathcal{S}$ at coordinate $(i, j)$. Let $Y_{m,n}$ be the element at location $(m, n)$ in the matrix of the interaction map corresponding to the image and the clicks in $\mathcal{S}$. $Y_{m,n}$ is calculated by:

$$Y_{m,n} = \min_{s_{i,j} \in \mathcal{S}} \sqrt{(m-i)^2 + (n-j)^2} \qquad (1)$$

In other words, the interaction maps are calculated using the minimum Euclidean distance between pixels and the user clicks. The pixel values in positive and negative interaction maps are truncated to 255. If no negative clicks are received, all pixel values in the negative interaction map are set to 255. Examples of positive and negative interaction maps are included in Fig. 2(a). For the postprocessing of the network output, we adopt a graph cut-based method similar to the one in [15].

## 4. Experiments

In this section, we show the experimental analysis and comparisons for the proposed method. Firstly, we describe the experimental settings. Then, we perform experimental analysis for the proposed TSLFN and MSRN. Finally, we make comparisons to state-of-the-art algorithms for interactive image segmentation.

### *4.1. Experimental settings*

**Datasets**. We conduct experiments on four datasets: Pascal VOC 2012 [32], Microsoft Coco [33], Grabcut [2] and Berkeley [34]. Pascal VOC 2012 and Microsoft Coco are benchmark datasets for object segmentation. Grabcut and Berkeley are benchmark datasets for interactive image segmentation. For Pascal VOC 2012, we employ its training set with 1464 images and validation set with 1449 images; for Microsoft Coco, we randomly select 20 images from each of its 80 categories, similarly to the setting in [15]; for Grabcut with 50 images and Berkeley with 100 images, we use all the images.

**Data partition**. We partition the data in the above datasets into training/validation/test data as follows. We use the training set of Pascal VOC 2012 as training/validation data. From the 1464 images, we randomly select 200 images as our validation data, and the rest images are used as our training data. We use the training data to train neural networks. We use the validation data to monitor and control the training process. Since it is practically too expensive to collect interaction data from real users for network training, we employ the method in [15] to generate synthetic user interactions for the objects in the training/validation data. We use the data apart from the training/validation data as the test data for performance evaluation (*i.e.* Pascal VOC 2012 validation set, Microsoft Coco, Grabcut, Berkeley).

**Performance evaluation**. We use intersection of union (IoU) of foreground to measure segmentation accuracy. It calculates the ratio of intersection between segmentation result and ground truth mask to the union of them. Based on IoU, we evaluate the performance of algorithms using two measures: foreground IoU vs. number of clicks, and number of clicks to achieve a certain foreground IoU. The former measure demonstrates the segmentation accuracy with respect to the number of user clicks; the latter measure shows the amount of user effort to achieve a certain segmentation accuracy. Given an object in an image, we automatically generate a sequence of clicks as user interactions (see below). We track foreground IoU vs. number of clicks, and we record the number of clicks to achieve a certain foreground IoU. For a dataset, we calculate the mean of each measure over all objects in this dataset. In this paper, we set 20 as the maximum number of clicks. For the second measure, if the certain IoU cannot be achieved in 20 clicks, we threshold the recorded number of clicks by 20.

**Generating click sequences**. We designed a method to automatically generate a sequence of clicks for a given object for performance evaluation. This method iterates between (1) adding a click into the click sequence based on the current segmentation result and (2) renewing the segmentation result using the updated click sequence. Given the current segmentation mask and the ground truth mask, we add a click as follows. Firstly, we find the false positive and the false negative regions in the current segmentation result. Then, we select the largest connected component among the false positive and the false negative regions. We place the click at the point which is within the selected region and farthest from the boundary of this region. This click is set to a positive click if it is placed at a false negative region, otherwise it is set to a negative click. After the click is added, we use the algorithm in evaluation to update the segmentation mask. The intuition of this method is that the added click focuses on the largest error region and it is placed in the central part of the region as much as possible.

### *4.2. Analysis of TSLFN*

Recall that our intuition to use a late fusion structure in TSLFN is to improve the impact of user interactions on the prediction result, as user interactions are more accurate information on the location of foreground/background. Therefore, as the first experiment in this subsection, we compare the impact of user clicks on the prediction result between two-stream and single-stream networks. The idea to measure the impact of user clicks on the prediction result is straightforward: a positive click has a higher impact if its surrounding region has higher response in the foreground probability map produced by the network; a negative click has a higher impact if its surrounding region has lower response in the foreground probability map. Therefore, if both positive and negative clicks achieve high impact on the network output, the responses around the two type of clicks in the foreground probability map should have well-separated distributions. Accordingly, the overall influence of positive and negative clicks can be measured by: (1) calculating the distributions of the responses in the regions around positive and negative clicks in the foreground probability map; (2) measuring how well the distributions corresponding to positive and negative clicks are separated. In this paper, we adopt decidability index (DI) to measure the degree of separation between the two distributions [35]:

$$\mathrm{DI} = \frac{|\mu_p - \mu_n|}{\sqrt{\frac{\sigma_p + \sigma_n}{2}}} \tag{2}$$

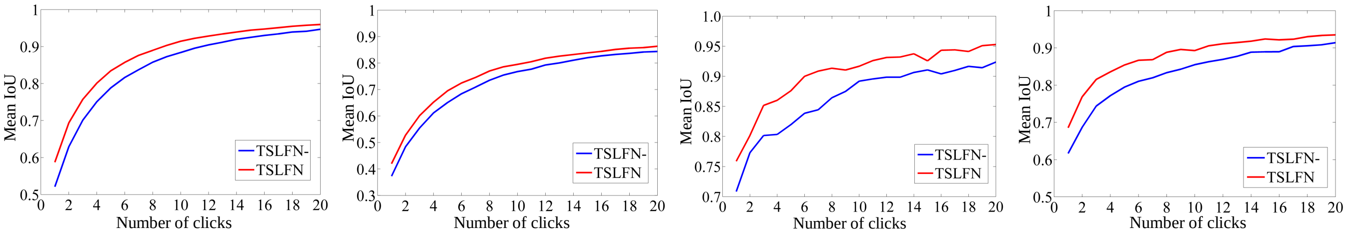
where $\mu_p$ and $\mu_n$ are the mean of the response distribution around positive and negative clicks, respectively; $\sigma_p$ and $\sigma_n$ denote the variance of the two distributions.

To analyse the impact of user clicks between two-stream and single-stream networks, we compare the DI between TSLFN and its single-stream modification. To modify the TSLFN to a single-stream network, we remove the interaction stream, and we concatenate the interaction maps with the image at the beginning of the network. Note that the TSLFN after this modification is equivalent to the single-stream fully convolutional network (SSFCN) in [15]. Thus, we refer to it as SSFCN. We measure the DI with 1, 5 and 10 user clicks automatically generated using the method in section 4.1 (we refer to this setting as free-choice). We calculate the DI based on the responses within a radius of 10 to positive and negative clicks in the probability map. If no negative clicks exist, we calculate the DI between the responses around the positive clicks and the responses in all background regions. We calculate the DI using the above methods for each object individually, and we report the mean DI on each of our test datasets.

In addition to the free-choice setting that allows the free choices of user clicks and leads to a combination of positive and negative clicks, we also study the impact of the clicks when only positive or negative clicks exist. This is to study the behaviour of the network for different types of user clicks. To study the case that only positive clicks exist, we force all the clicks to be put on the foreground (referred to as all-positive); we measure the DI between the regions around the positive clicks and the whole background region. To investigate the case that only negative clicks exist, we consider an approximate setting: we force the first click to be on the foreground and the rest clicks to be on the background (referred to as single-positive); and we measure the DI between the regions around the negative clicks and the whole foreground region. The reason to use an approximate setting is that all of our training data have at least one positive click given the method to generate them (see section 4.1), hence the network is not well trained to handle the data without positive clicks.

Tab. 1 shows the DIs on the four datasets with the user click settings of free-choice, all-positive and single-positive. We can see that TSLFN has a consistently higher DI compared to SSFCN. This means that the distributions of the responses around positive and negative clicks are more separated in the probability maps of TSLFN. In other words, the user clicks have higher impact with the two-stream network structure. This is consistent with our intuition of using a two-stream network. Also, we find that this trend holds for all three settings of user clicks. This shows that the improvement on the impact of user clicks achieved by the two-stream network is consistent for different types of clicks.

Since the DIs calculated as above are based on the regions around the positive and negative user clicks, they do not represent the separability of the responses between the whole foreground and background regions; hence, they do not represent how good the final segmentation results are. To validate if the higher impact of user clicks benefits the final segmentation performance, we also need to compare the segmentation accuracy between TSLFN and SSFCN. Note that, in the rest of the paper, we do not restrict the



(a) Pascal VOC 2012 (b) Microsoft Coco (c) Grabcut (d) Berkeley

Figure 3: Mean IoU vs. number of clicks for the analysis of the effect of deep features from the interaction stream of the TSLFN

Table 1: Decidability index (DI) between regions around positive and negative clicks in the probability map (best performance in bold for each setting)

| Dataset | Free-choice | | All-positive | | Single-positive | |
|---|---|---|---|---|---|---|
| | SSFCN | TSLFN | SSFCN | TSLFN | SSFCN | TSLFN |
| Pascal VOC 2012 (1 click) | 12.97 | **15.60** | 12.97 | **15.60** | - | - |
| Microsoft Coco (1 click) | 16.79 | **20.37** | 16.79 | **20.37** | - | - |
| Grabcut (1 click) | 19.38 | **21.38** | 19.38 | **21.38** | - | - |
| Berkeley (1 click) | 71.42 | **90.91** | 71.42 | **90.91** | - | - |
| Pascal VOC 2012 (5 clicks) | 2.57 | **6.40** | 2.79 | **4.03** | 2.16 | **2.92** |
| Microsoft Coco (5 clicks) | 2.72 | **9.61** | 2.40 | **3.58** | 1.29 | **2.10** |
| Grabcut (5 clicks) | 3.84 | **7.91** | 3.14 | **4.66** | 1.89 | **3.04** |
| Berkeley (5 clicks) | 2.85 | **5.22** | 3.05 | **4.33** | 1.33 | **2.23** |
| Pascal VOC 2012 (10 clicks) | 1.24 | **2.33** | 2.33 | **3.20** | 2.34 | **3.03** |
| Microsoft Coco (10 clicks) | 1.20 | **2.37** | 2.06 | **2.87** | 1.30 | **1.89** |
| Grabcut (10 clicks) | 1.53 | **3.16** | 2.15 | **3.09** | 1.94 | **3.06** |
| Berkeley (10 clicks) | 1.33 | **2.46** | 2.03 | **2.88** | 1.50 | **2.21** |

types of user clicks (*i.e.* we follow the free-choice setting above). This is for two reasons. First, it is the most general case to allow users to freely place positive and negative clicks. Second, the free-choice setting actually covers the all-positive and single-positive settings; for example, when the foreground is very small, the free-choice setting is very likely to produce a click sequence the same as the click sequence produced by the single-positive setting. As shown in Fig. 4, Tab. 2 and Tab. 3, TSLFN has a better final segmentation performance compared to SSFCN. This observation suggests that an improved performance is indeed achieved by improving the impact of user interactions on the network output with a two-stream network architecture.

Another interesting observation from Tab. 1 is that the DIs generally drop as the number of user clicks increases. We think there are two possible reasons. First, with a single user click, the network may be more focused around this only click; this leads to very large difference between the responses around the click and the responses in the background, so it leads to a large DI. In contrast, with more user clicks, the network may try to achieve a trade-off between the influence of all clicks; this may lead to decreased responses around each click and hence a lower DI. Second, we find that the main object can be generally segmented with high accuracy with very few clicks (see Tab. 2). As a result, with 5 or 10 user clicks, there may be many clicks near to object boundary. The responses around these clicks may lower the overall DI, as the responses in the foreground probability maps may be weaker around the object boundary and hence they are less separating between positive and negative clicks.

Moreover, closely examining the results in Tab. 1, we can find that the above observation on the decreasing DI with respect to the increasing number of user clicks only holds for the free-choice and all-positive settings. For the single-positive setting, the DI is very similar between 5 and 10 clicks. This observation leads to a further interesting possibility on the behaviour of the network: the network treats the positive clicks competitively, while it treats the negative clicks equally. On the one hand, with the all-positive setting, the DIs decrease when the number of clicks increases. This may mean that there exists a competition between the impact of each individual positive click; it leads to a trade-off on the impact of each positive click and this lowers the overall DI with more positive clicks. On the other hand, with the single-positive setting, the DIs are very similar between 5 and 10 clicks. This may mean that adding negative clicks changes little on the impact of each individual negative click. In other words, the network treats each negative click equally.

The above experiments validate our idea that the two-stream network allows the user clicks to have a higher impact on the prediction result, and it leads to improved performance compared to the single-stream network. However, this leads to another question: do we need deep features of user interactions to apply this impact? In other words, is the interaction stream necessary for the TSLFN? To justify the effect of the interaction stream that produces deep features for user interactions, we compare the performance between two networks: (1) TSLFN; (2) TSLFN with the interactive stream removed (referred to as TSLFN-). Specifically, in TSLFN-, the interaction maps are resized and concatenated with the features at the end of the image stream; the concatenated features are then used as the input of the fusion net to predict the foreground (note that this is different from SSFCN where the interaction maps are concatenated with the original input image at the beginning of the whole network). Fig. 3 compares the performance between TSLFN- and TSLFN (note that the results in this figure are based on the stride-32 networks; the performance on the final stride-8 networks are likely to be similar, as the stride-8 networks are based on the stride-32 networks). It can be seen that the performance drops for TSLFN-. This result shows that the deep features of user interactions from the interaction stream is also important for the TSLFN to achieve a good performance. This may be because deep features provide a richer and more meaningful representation of user clicks, and it can more accurately guide the segmentation process when fused with image features.

Finally, we report some experimental results related to the design of the proposed TSLFN. As discussed at the end of section 3.1, we could use different depth in the image/interaction streams and the fusion net to construct TSLFN, given the VGG16 base network. Specifically, the VGG16 base network has 5 Conv-ReLU-Pool (CRP) blocks. Our TSLFN structure in Fig. 2(a) uses the first 4 CRP blocks to form the image/interaction streams, and it uses the rest part of VGG16 as the fusion net. One can create variations of this architecture by using a different number of CRB blocks to form the image/interaction streams. This leads to variations of the proposed TSLFN with different depth in the image/interaction streams and the fusion net. We use TSLFN_$i$ to denote the variation of the proposed TSLFN with the first $i$ CRP blocks in the base network used as the image/interaction streams. The proposed TSLFN shown in Fig. 2(a) is essentially equivalent to TSLFN_4. It has four variations: TSLFN_1, TSLFN_2, TSLFN_3, TSLFN_5. Among these variations, TSLFN_1 has the shallowest image/interaction streams and the deepest fusion net, while TSLFN_5 has the deepest image/interaction streams and the shallowest fusion net.

Fig. 5, Tab. 4 and Tab. 5 show the performance of all the above variations of the proposed TSLFN. It can be seen that the proposed TSLFN (TSLFN_4 in Fig. 5, Tab. 4 and Tab. 5) generally has the highest performance among its variations. A possible reason is as the one we discussed at the end of section 3.1: there exists a trade-off between the impact of user interactions and the prediction capacity with different depths in image/interaction streams and fusion net; the proposed TSLFN structure as shown in Fig. 2(a) achieves the best trade-off between the two factors compared to its other variations, given our base network.

In this subsection we analysed the proposed TSLFN. The results confirm that: (1) compared to the single-stream network, the two-stream structure of the TSLFN allows the information from user clicks to have a higher impact on the network output, and it leads to better performance; (2) extracting deep features from user interactions is also important for the TSLFN to achieve a better performance. We also validated the design choice of the proposed TSLFN, showing that it generally achieves the best performance among its variations with the given base network.

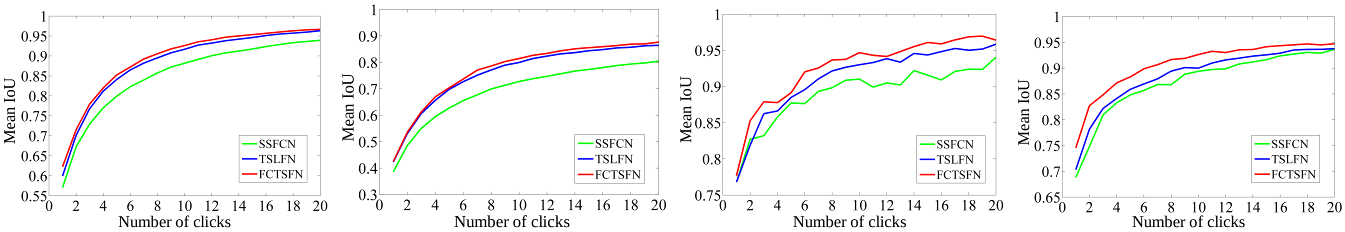
### *4.3. Analysis of MSRN*

To analyse the effect of the MSRN, we compare the performance between two networks: TSLFN and FCTSFN (*i.e.* TSLFN+MSRN). By comparing the performance reported in Fig. 4, Tab. 2 and Tab. 3, we can see that FCTSFN has a consistently better performance than TSLFN. These observations validate the effectiveness of the MSRN to utilize multi-scale features to refine the segmentation result. In our opinion, two possible reasons lead to the improved performance. First, MSRN makes prediction at full resolution, hence it is more accurate at object boundaries. Second, MSRN utilizes features from the beginning to the end of the network. Therefore, it fuses information from low-level features such as colors/boundaries to high-level features with object-level understanding; this allows the network to build a more comprehensive understanding on the foreground and background, and it leads to more accurate segmentation results.

### *4.4. Comparison with existing algorithms*

In this subsection, we compare the proposed network to state-of-the-art algorithms. We divide our comparisons into two categories: restricted comparison and unrestricted comparison. In the restriction comparison, we conduct experiments strictly under our experimental setting in section 4.1; we either run the source codes of the comparison methods or implement the comparison methods by ourselves. In the unrestricted comparison, we directly compared with the performance measure cited from published papers. Note that, in the unrestricted comparison, the results are not fully comparable due to the differences in the experimental setting in different papers. However, it shows the performance of the proposed network among state-of-the-art methods with open choices for experimental settings.

**Restricted comparison.** For the restricted comparison, we compare to the following methods: graph cut



(a) Pascal VOC 2012 (b) Microsoft Coco (c) Grabcut (d) Berkeley

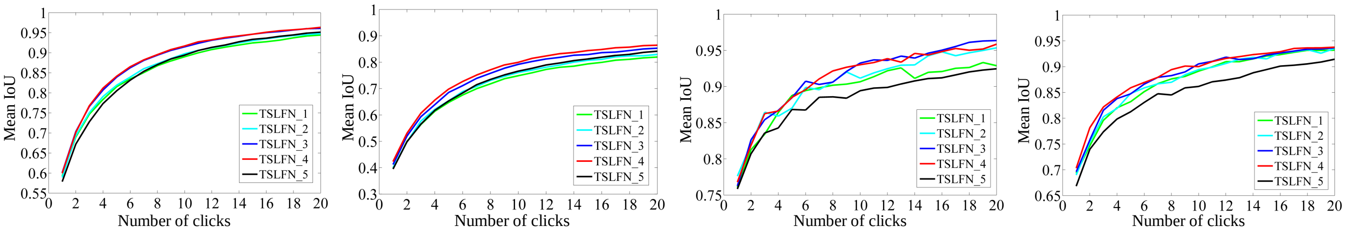
Figure 4: Mean IoU vs. number of clicks for the analysis of the TSLFN and MSRN

Table 2: Mean number of clicks to achieve a certain IoU for the analysis of the TSLFN and MSRN (best performance in bold)

| Dataset | SSFCN | TSLFN | FCTSFN |
|---|---|---|---|
| Pascal VOC 2012 (85% IoU) | 5.81 | 4.95 | **4.58** |
| Microsoft Coco (85% IoU) | 11.42 | 9.97 | **9.62** |
| Grabcut (90% IoU) | 5.02 | 4.28 | **3.76** |
| Berkeley (90% IoU) | 8.48 | 7.89 | **6.49** |

Table 3: Mean IoU at certain number of clicks for the analysis of the TSLFN and MSRN (in percentage, best performance in bold)

| Dataset | SSFCN | TSLFN | FCTSFN |
|---|---|---|---|
| Pascal VOC 2012 (1 click) | 57.0 | 60.0 | **62.3** |
| Microsoft Coco (1 click) | 38.6 | 42.3 | **42.5** |
| Grabcut (1 click) | 76.8 | 76.8 | **77.7** |
| Berkeley (1 click) | 68.8 | 70.3 | **74.5** |
| Pascal VOC 2012 (3 clicks) | 73.0 | 76.8 | **78.0** |
| Microsoft Coco (3 clicks) | 55.1 | 60.7 | **61.2** |
| Grabcut (3 clicks) | 83.2 | 86.3 | **87.9** |
| Berkeley (3 clicks) | 81.0 | 82.2 | **84.8** |
| Pascal VOC 2012 (10 clicks) | 88.2 | 91.7 | **92.6** |
| Microsoft Coco (10 clicks) | 72.8 | 80.0 | **81.5** |
| Grabcut (10 clicks) | 91.0 | 93.0 | **94.7** |
| Berkeley (10 clicks) | 89.4 | 90.0 | **92.6** |



(a) Pascal VOC 2012 (b) Microsoft Coco (c) Grabcut (d) Berkeley

Figure 5: Mean IoU vs. number of clicks for the analysis of the design of the proposed TSLFN

(GC) [1], geodesic matting (GM) [3], random walk (R-W) [7], Euclidean star convexity (ESC) [5], geodesic star convexity (GSC) [5], single stream FCN (SSFCN) [15].

Fig. 6 shows the mean IoU vs. number of clicks for all comparison methods for all the four datasets on the test data. Tab. 6 shows the mean IoU at some certain number of clicks (1, 3, 10). It can be seen that the proposed FCTSFN achieves improved performance compared to the other methods. Specifically, on the Pascal VOC 2012, Microsoft Coco and Berkeley datasets, FCTSFN performs better compared to the other methods. On the Grabcut dataset, FCTSFN achieves better performance when the

Table 4: Mean number of clicks to achieve a certain IoU for the analysis of the design of the proposed TSLFN (best performance in bold)

| Dataset | TSLFN_1 | TSLFN_2 | TSLFN_3 | TSLFN_4 | TSLFN_5 |
|---|---|---|---|---|---|
| Pascal VOC 2012 (85% IoU) | 5.63 | 5.43 | **4.95** | **4.95** | 5.73 |
| Microsoft Coco (85% IoU) | 11.06 | 10.83 | 10.18 | **9.97** | 10.99 |
| Grabcut (90% IoU) | 4.60 | 4.66 | 4.44 | **4.28** | 5.14 |
| Berkeley (90% IoU) | 8.21 | 8.57 | 8.15 | **7.89** | 9.27 |

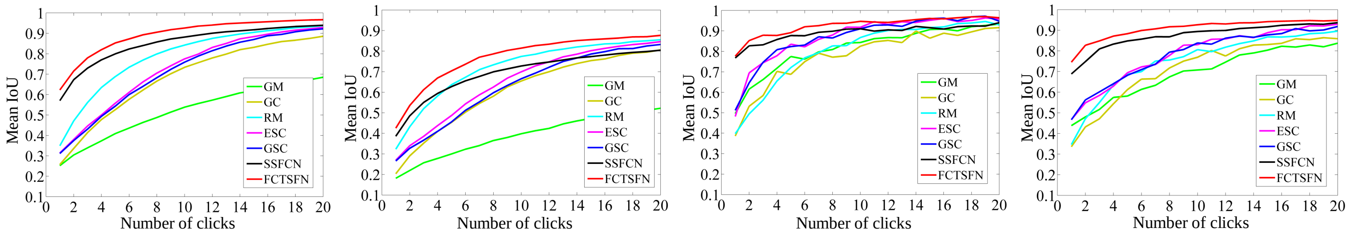
Table 5: Mean IoU at certain number of clicks for the analysis of the design of the proposed TSLFN (in percentage, best performance in bold)

| Dataset | TSLFN_1 | TSLFN_2 | TSLFN_3 | TSLFN_4 | TSLFN_5 |
|---|---|---|---|---|---|
| Pascal VOC 2012 (1 click) | 59.3 | 58.9 | **60.1** | 60.0 | 57.9 |
| Microsoft Coco (1 click) | 39.7 | 40.0 | 41.2 | **42.3** | 39.5 |
| Grabcut (1 click) | 76.1 | **77.6** | 76.3 | 76.8 | 75.9 |
| Berkeley (1 click) | 69.3 | 69.0 | 69.6 | **70.3** | 66.8 |
| Pascal VOC 2012 (3 clicks) | 74.7 | 75.1 | 76.7 | **76.8** | 72.8 |
| Microsoft Coco (3 clicks) | 56.4 | 57.7 | 59.3 | **60.7** | 56.5 |
| Grabcut (3 clicks) | 83.4 | **86.4** | 85.4 | 86.3 | 83.6 |
| Berkeley (3 clicks) | 79.6 | 80.3 | 81.6 | **82.2** | 77.4 |
| Pascal VOC 2012 (10 clicks) | 89.0 | 89.8 | 91.5 | **91.7** | 89.4 |
| Microsoft Coco (10 clicks) | 74.9 | 76.1 | 79.2 | **80.0** | 76.6 |
| Grabcut (10 clicks) | 90.7 | 91.2 | **93.2** | 93.0 | 89.4 |
| Berkeley (10 clicks) | 89.1 | 89.4 | **90.6** | 90.0 | 86.2 |

number of clicks is lower than 10; when the number of clicks is larger than 10, FCTSFN performs similarly to ESC and GSC, and it has better performance compared to the other methods. The proposed FCTSFN shows a larger advantage on the Pascal VOC 2012, Microsoft Coco and Berkeley datasets than on the Grabcut dataset. One possible reason is that the Grabcut dataset has a smaller number of images with more distinct foreground/background. Therefore, FCTSFN performs similarly to ESC and GSC on the Grabcut dataset given sufficient number of clicks. In summary, the proposed FCTSFN shows consistently improved performance on larger and more challenging datasets, while it still achieves stable and top performance on smaller and less challenging datasets. Tab. 7 reports the mean number of clicks to achieve a certain IoU. It can be seen that the proposed FCTSFN needs the least number of clicks on all the datasets. Note that the best possible number of clicks to achieve a certain IoU is 1. Therefore, the proposed FCTSFN achieves an improvement of $(5.81 - 4.58)/(5.81 - 1) \approx 25.6\%$ towards the best possible performance with respect to SSFCN on the VOC 2012 dataset. This figure is 23.3%, 31.3% and 26.6% for the Microsoft Coco, Grabcut and Berkeley datasets, respectively. Fig. 7 shows some example results of different methods given the same user clicks. Fig. 8 shows some example results of the proposed method on different objects in the test datasets, with automatically generated click sequences with up to 5 clicks.

**Unrestricted comparison.** For the unrestricted comparison, we compare with the following methods: RIS-Net [20], DEXTR [19] and latent diversity network (LDN) [28]. We directly cite the number of clicks to achieve a certain IoU reported in these papers. Note that, as discussed above, these results are not directly comparable due to the differences in the experiment settings. For example, for the test data on Microsoft Coco dataset, different methods use different random sampling settings; these methods also adopt different training data and various training strategies; *etc.* However, this comparison shows the performance of the proposed network among state-of-the-art methods with open choices for experimental settings.

Tab. 8 reports the performance of all methods in unrestricted comparison. It can be seen that the proposed FCTSFN achieves competitive performance on Pascal VOC 2012, Grabcut and Berkeley datasets. Specifically, compared to RIS-Net, FCTSFN needs fewer clicks to achieve a certain IoU on Pascal VOC 2012 and Grabcut datasets; it needs 0.46 more clicks than RIS-Net to achieve 90% IoU on Berkeley dataset. Compared to DEXTR, FCTSFN performs better on Grabcut dataset, and it needs 0.58 more clicks to achieve a 85% IoU on Pascal VOC 2012 dataset. Compared to LDN, FCTSFN achieves a better performance on Grabcut dataset. Note that, compared to the proposed FCTSFN, DEXTR achieves the reported performance with more training data (Pascal VOC 2012 + SBD [36]), with an online hard example mining (OHEM) [37] based training strategy and a more advanced base network (ResNet-101 [10]); similarly, LDN achieve its



(a) Pascal VOC 2012 (b) Microsoft Coco (c) Grabcut (d) Berkeley

Figure 6: Mean IoU vs. number of clicks for restricted comparisons

Table 6: Mean IoU at certain number of clicks for restricted comparisons (in percentage, best performance in bold)

| Dataset | GC | GM | RW | ESC | GSC | SSFCN | FCTSFN |
|---|---|---|---|---|---|---|---|
| Pascal VOC 2012 (1 click) | 25.2 | 25.7 | 34.9 | 31.4 | 31.2 | 57.0 | **62.3** |
| Microsoft Coco (1 click) | 18.1 | 20.3 | 32.3 | 26.9 | 26.6 | 38.6 | **42.5** |
| Grabcut (1 click) | 49.6 | 38.6 | 39.8 | 48.2 | 51.2 | 76.8 | **77.7** |
| Berkeley (1 click) | 43.8 | 33.6 | 34.6 | 46.7 | 46.7 | 68.8 | **74.5** |
| Pascal VOC 2012 (3 clicks) | 33.9 | 41.4 | 56.1 | 44.6 | 43.1 | 73.0 | **78.0** |
| Microsoft Coco (3 clicks) | 25.7 | 35.2 | 52.0 | 38.5 | 36.7 | 55.1 | **61.2** |
| Grabcut (3 clicks) | 66.2 | 58.4 | 56.1 | 74.4 | 74.6 | 83.2 | **87.9** |
| Berkeley (3 clicks) | 51.6 | 47.1 | 55.0 | 58.4 | 60.1 | 81.0 | **84.8** |
| Pascal VOC 2012 (10 clicks) | 53.8 | 73.3 | 84.2 | 77.5 | 75.9 | 88.2 | **92.6** |
| Microsoft Coco (10 clicks) | 39.8 | 65.4 | 77.4 | 69.9 | 66.6 | 72.8 | **81.5** |
| Grabcut (10 clicks) | 84.3 | 82.5 | 86.8 | 91.8 | 91.0 | 91.0 | **94.7** |
| Berkeley (10 clicks) | 70.8 | 76.9 | 80.6 | 83.0 | 83.8 | 89.4 | **92.6** |

Table 7: Mean number of clicks to achieve a certain IoU for resrticted comparisons (best performance in bold)

| Dataset | GC | GM | RW | ESC | GSC | SSFCN | FCTSFN |
|---|---|---|---|---|---|---|---|
| Pascal VOC 2012 (85% IoU) | 14.81 | 10.59 | 7.98 | 8.22 | 8.48 | 5.81 | **4.58** |
| Microsoft Coco (85% IoU) | 17.74 | 14.57 | 11.71 | 11.70 | 12.11 | 11.42 | **9.62** |
| Grabcut (90% IoU) | 9.70 | 9.26 | 10.28 | 5.84 | 5.02 | 5.02 | **3.76** |
| Berkeley (90% IoU) | 13.68 | 14.10 | 13.46 | 9.73 | 9.38 | 8.48 | **6.49** |

Table 8: Mean number of clicks to achieve a certain IoU for unresrticted comparisons (best performance in bold)

| Dataset | RIS-Net | DEXTR | LDN | FCTSFN |
|---|---|---|---|---|
| Pascal VOC 2012 (85% IoU) | 5.12 | **4.00** | - | 4.58 |
| Microsoft Coco (85% IoU) | - | - | **7.89** | 9.62 |
| Microsoft Coco seen categories (85% IoU) | 5.98 | - | - | - |
| Microsoft Coco unseen categories (85% IoU) | 6.44 | - | - | - |
| Grabcut (90% IoU) | 5.00 | 4.00 | 4.79 | **3.76** |
| Berkeley (90% IoU) | **6.03** | - | - | 6.49 |

reported performance with a larger training set (SBD) and a more advanced segmentation network (context aggregation network [38, 39]). In contrast, the proposed FCTSFN achieves the performance in Tab. 8 with less training data (Pascal VOC 2012 only), without hard mining on training data during the training process, and with a less advanced base network (VGG16).

On the other hand, it can be seen from Tab. 8 that the proposed FCTSFN needs the most number of clicks to achieve an IoU of 85% on Microsoft Coco dataset compared to RIS-Net and LDN. However, since each comparison algorithm adopts different random sampling settings,

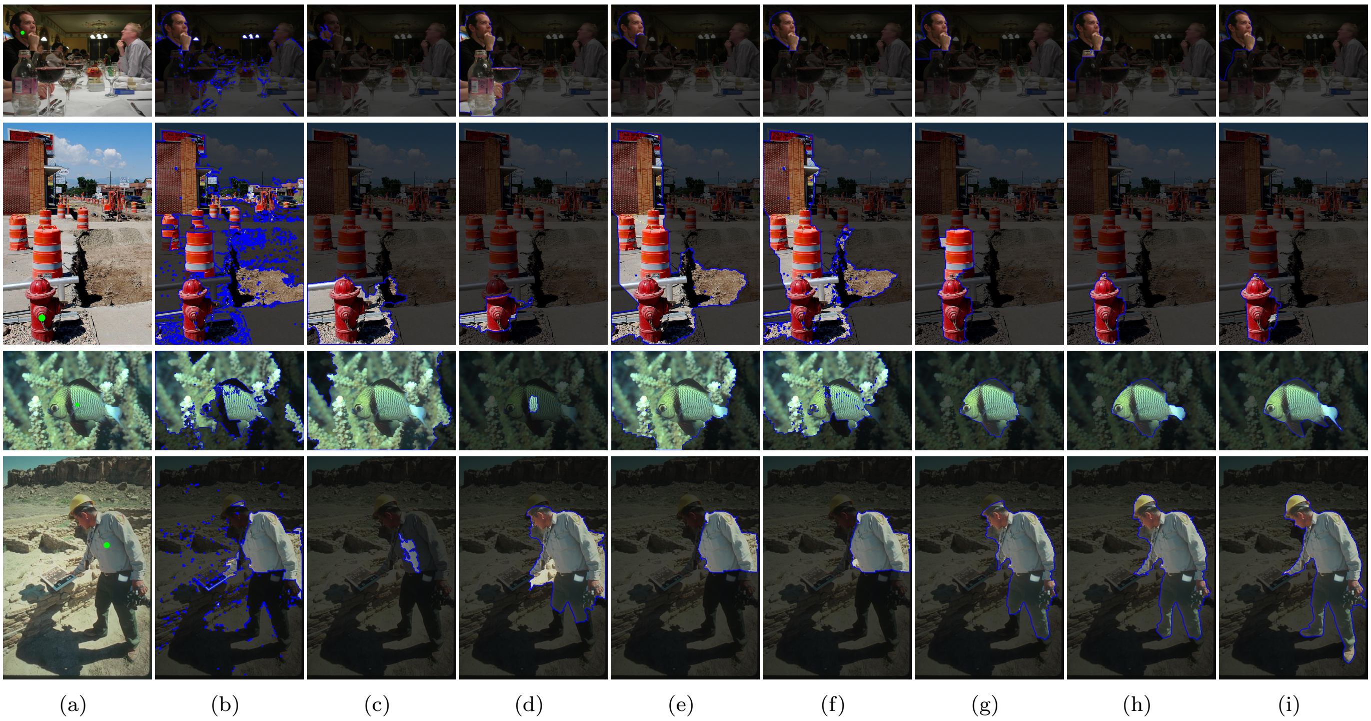


Figure 7: Examples of segmentation results of different methods given the same user interaction. (a) Original image with user clicks; (b) GC; (c) GM; (d) RW; (e) ESC; (f) GSC; (g) SSFCN; (h) FCTSFN; (i) Ground truth

we cannot estimate the effect of such a sampling on the final performance. For example, our implementation of SSFCN on our Microsoft Coco test data (as shown in Tab. 7) results in lower performance than those reported in other implementations with other randomly sampled Microsoft Coco test data [20, 28].

In this subsection, we showed that the proposed FCTSFN achieved improved performance compared to other comparison methods in restricted comparisons. In unrestricted comparisons, it also achieves competitive performance with less training data and with a less advanced based network compared to other methods.

## 5. Conclusions

In this paper, we proposed a novel fully convolutional two-stream fusion network (FCTSFN) for interactive image segmentation. The intention is to firstly use a two-stream late fusion network (TSLFN) to allow the user interactions to have more direct and higher impact on the segmentation results to achieve improved accuracy, then use a multi-scale refining network (MSRN) to refine the segmentation result at full resolution to address the resolution loss in TSLFN. We conduct comprehensive experimental analysis and comparisons on four benchmark datasets. The main findings are summarised as follows:

- We experimentally validate that the two-stream structure in TSLFN allows the user interactions to have a higher impact on the segmentation results and it achieves improved performance compared to single-stream networks.
- We experimentally validate the significance of the interaction stream in the TSLFN: the TSLFN with the interaction stream performs better than the TSLFN without this stream. This means that the interaction stream in the proposed network successfully learn richer and more meaningful feature representations from individual user interaction data.
- We experimentally validate the design choice of the proposed TSLFN. We show that the proposed architecture achieves generally better performance compared to its variations, given the fixed base network.
- We experimentally validate that the foreground refining performed by the MSRN in the FCTSFN leads to a further improvement on the performance of the TSLFN.
- In restricted comparisons, the proposed FCTSFN achieves better performance compared to state-of-the-art methods.
- In unrestricted comparisons, the proposed FCTSFN also achieves competitive performance with less training data and a less advanced base network, compared to state-of-the-art methods.

Future works may focus on: (1) implementing the two-stream structure with more advanced base networks to

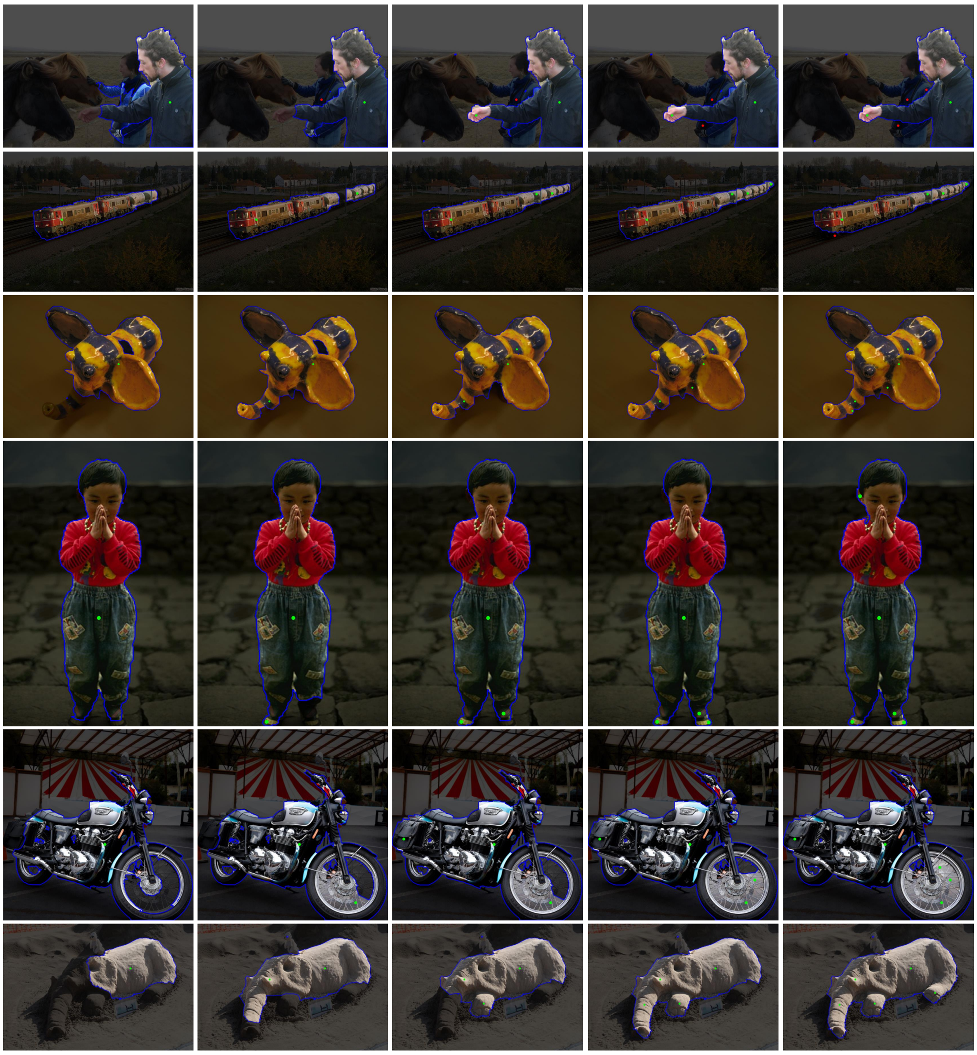


Figure 8: Examples of segmentation results of the proposed method on different objects in the test data with automatically generated click sequences; from left to right in each row: the number of clicks increases from 1 to 5.

achieve better performance; (2) conducting more experimental and theoretical analysis to gain a deeper insight into the two-stream network structure for interactive image segmentation.

## Acknowledgments

This work was supported by Ice Communication Limited and Innovate UK (project KTP/10412).

## References

[1] Y. Y. Boykov, M. P. Jolly, Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images, in: Proceedings of 8th IEEE International Conference on Computer Vision (ICCV), Vol. 1, 2001, pp. 105–112.

[2] C. Rother, V. Kolmogorov, A. Blake, "GrabCut": interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics (TOG) 23 (3) (2004) 309–314.

[3] X. Bai, G. Sapiro, A geodesic framework for fast interactive image and video segmentation and matting, in: Proceedings of 11th IEEE International Conference on Computer Vision (ICCV), 2007, pp. 1–8.

[4] B. Price, B. Morse, S. Cohen, Geodesic graph cut for interactive image segmentation, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3161–3168.

[5] V. Gulshan, C. Rother, A. Criminisi, A. Blake, A. Zisserman, Geodesic star convexity for interactive image segmentation, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3129–3136.

[6] V. Vezhnevets, V. Konouchine, "GrowCut" - Interactive multi-label N-D image segmentation by cellular automata, in: Proceedings of Graphicon, 2005, pp. 150–156.

[7] L. Grady, Random walks for image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 1768–1783.

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[9] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of International Conference on Learning Representations (ICLR), 2015, `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`.

[10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for im-

age recognition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[11] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (4) (2017) 640–651.

[12] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P. H. S. Torr, Conditional random fields as recurrent neural networks, in: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1529–1537.

[13] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[14] Y. Li, H. Qi, J. Dai, X. Ji, Y. Wei, Fully convolutional instance-aware semantic segmentation, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[15] N. Xu, B. Price, S. Cohen, J. Yang, T. Huang, Deep interactive object selection, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 373–381.

[16] G. Wang, M. A. Zuluaga, W. Li, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, T. Vercauteren, DeepIGeoS: a deep interactive geodesic framework for medical image segmentation, arXiv preprint arXiv:1707.00652v2.

[17] A. S. Boroujerdi, M. Khanian, M. Breuß, Deep interactive region segmentation and captioning, arXiv preprint arXiv:1707.08364.

[18] N. Xu, B. Price, S. Cohen, J. Yang, T. Huang, Deep GrabCut for object selection, arXiv preprint arXiv:1707.00243.

[19] K. K. Maninis, S. Caelles, J. Pont-Tuset, L. Van Gool, Deep extreme cut: From extreme points to object segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[20] J. H. Liew, Y. Wei, W. Xiong, S. H. Ong, J. Feng, Regional interactive image segmentation networks, in: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2017.

[21] S. Mahadevan, P. Voigtlaender, B. Leibe, Iteratively trained interactive segmentation, arXiv preprint arXiv:1805.04398.

[22] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239.

[23] V. Kolmogorov, R. Zabin, What energy functions can be minimized via graph cuts?, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 147–159.

[24] J. V. Neumann, Theory of self-reproducing automata, University of Illinois Press, 1966.

[25] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceedings of European Conference on Computer Vision (ECCV), 2014, pp. 818–833.

[26] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1717–1724.

[27] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[28] Z. Li, Q. Chen, V. Koltun, Interactive image segmentation with latent diversity, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[29] C. Hazirbas, L. Ma, C. Domokos, D. Cremers, FuseNet: incorporating depth into semantic segmentation via fusion-based CNN architecture, in: Asian Conference on Computer Vision (ACCV), 2016.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.

[31] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, in: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1529–1537.

[32] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, International Journal of Computer Vision 88 (2) (2010) 330–338.

[33] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: common objects in context, in: Proceedings of European Conference on Computer Vision (ECCV), 2014, pp. 740–755.

[34] K. McGuinness, N. E. OConnor, A comparative evaluation of interactive segmentation algorithms, International Journal of Computer Vision 43 (2) (2010) 434–444.

[35] J. Daugman, How iris recognition work, IEEE Transactions on Circuits and Systems for Video Technology 14 (1) (2004) 21–30.

[36] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, J. Malik, Semantic contours from inverse detectors, in: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2011.

[37] A. Shrivastava, A. Gupta, R. Girshick, Training region based object detectors with online hard example mining, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[38] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: Proceedings of International Conference on Learning Representations (ICLR), 2016.

[39] Q. Chen, J. Xu, V. Koltun, Fast image processing with fully-convolutional networks, in: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2017.