Richer Convolutional Features for Edge Detection

Yun Liu[1] Ming-Ming Cheng[1] * Xiaowei Hu[1] Kai Wang[1] Xiang Bai[2]
[1]Nankai University [2]HUST
http://mmcheng.net/rcfedge/

Abstract

*In this paper, we propose an accurate edge detector using richer convolutional features (RCF). Since objects in natural images possess various scales and aspect ratios, learning the rich hierarchical representations is very critical for edge detection. CNNs have been proved to be effective for this task. In addition, the convolutional features in CNNs gradually become coarser with the increase of the receptive fields. According to these observations, we attempt to adopt richer convolutional features in such a challenging vision task. The proposed network fully exploits multiscale and multilevel information of objects to perform the image-to-image prediction by combining all the meaningful convolutional features in a holistic manner. Using VGG16 network, we achieve state-of-the-art performance on several available datasets. When evaluating on the well-known BSDS500 benchmark, we achieve ODS F-measure of **0.811** while retaining a fast speed (**8** FPS). Besides, our fast version of RCF achieves ODS F-measure of **0.806** with **30** FPS.*

1. Introduction

Edge detection, which aims to extract visually salient edges and object boundaries from natural images, has remained as one of the main challenges in computer vision for several decades. It is usually considered as a low-level technique, and varieties of high-level tasks have greatly benefited from the development of edge detection, such as object detection [17, 55], object proposal [9, 54, 60–62] and image segmentation [1, 3, 8, 56].

Typically, traditional methods first extract local cues of brightness, colors, gradients and textures, or other manually designed features like Pb [40], gPb [2], and Sketch tokens [36], then sophisticated learning paradigms [14, 57] are used to classify edge and non-edge pixels. Although edge detection approaches using low-level features have made great improvement in these years [33], their limitations are also obvious. For example, edges and boundaries are often defined to be semantically meaningful, however, it is difficult to use low-level cues to represent object-level information. Under these circumstances, gPb [2] and Structured Edges [14] try to use complex strategies to capture global features as much as possible.

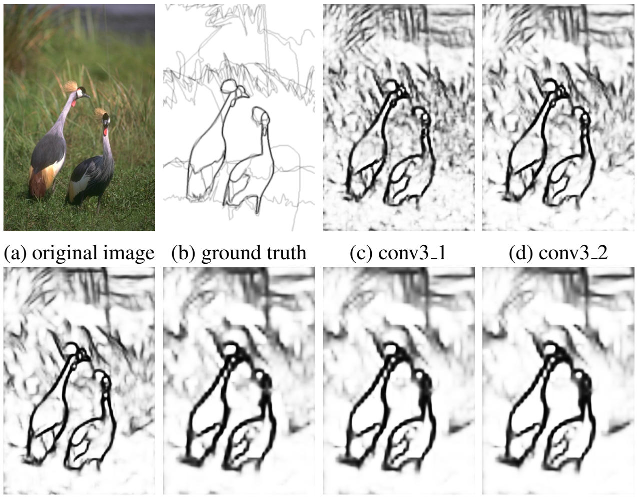
(a) original image (b) ground truth (c) conv3_1 (d) conv3_2
(e) conv3_3 (f) conv4_1 (g) conv4_2 (h) conv4_3

Figure 1: We build a simple network based on VGG16 [50] to produce side outputs of *conv3_1*, *conv3_2*, *conv3_3*, *conv4_1*, *conv4_2* and *conv4_3*. One can clearly see that convolutional features become coarser gradually, and the intermediate layers *conv3_1*, *conv3_2*, *conv4_1*, and *conv4_2* contain lots of useful fine details that do not appear in other layers.

In the past few years, convolutional neural networks (CNNs) have become popular in the computer vision community by substantially advancing the state-of-the-art of various tasks, including image classification [31, 50, 52], object detection [20, 21, 34, 43] and semantic segmentation [7, 38] *etc*. Since CNNs have a strong capability to learn high-level representations of natural images automatically, there is a recent trend of using convolutional networks to perform edge detection. Some well-known CNN-based methods have pushed forward this field significantly, such as DeepEdge [4], N^4-Fields [19], CSCNN [26], DeepCon-

[*] M.M. Cheng (cmm@nankai.edu.cn) is the corresponding author.

tour [47], and HED [58]. Our algorithm falls into this category as well.

To see the information obtained by different convolution (*i.e. conv*) layers in edge detection, we build a simple network to produce side outputs of intermediate layers using VGG16 [50] which has five *conv* stages. Fig. 1 shows an example. We discover that convolutional features become coarser gradually and intermediate layers contain lots of useful fine details. On the other hand, since richer convolutional features are highly effective for many vision tasks, many researchers make efforts to develop deeper networks [25]. However, it is difficult to get the networks to converge when going deeper because of vanishing/exploding gradients and training data shortage (*e.g.* for edge detection). So why don't we make full use the CNN features we have now? Our motivation is based on these observations. Unlike previous CNN methods, the proposed novel network uses the CNN features of all the *conv* layers to perform the pixel-wise prediction in an image-to-image fashion, and thus is able to obtain accurate representations for objects or object parts in different scales. Concretely speaking, we attempt to utilize the CNN features from all the *conv* layers in a unified framework that can be potentially generalized to other vision tasks. By carefully designing a universal strategy to combine hierarchical CNN features, our system performs very well in edge detection.

When evaluating the proposed method on BSDS500 dataset [2], we achieve the best trade-off between effectiveness and efficiency with the ODS F-measure of **0.811** and the speed of **8** FPS. It even outperforms the result of human perception (ODS F-measure 0.803). In addition, the fast version of RCF is also presented, which achieves ODS F-measure of **0.806** with **30** FPS.

2. Related Work

Since edge detection was set as one of the most fundamental problems in computer vision [15,18,46], researchers have struggled on it for nearly 50 years, and there have emerged a large number of materials. Broadly speaking, we can roughly categorize these approaches into three groups: early pioneering ones, learning based ones using handcrafted features and deep learning based ones. Here we briefly review some representative approaches that were developed in the past few decades.

Early pioneering methods mainly focused on the utilization of intensity and color gradients. Robinson [46] discussed a quantitative measure in choosing color coordinates for the extraction of visually significant edges and boundaries. [39, 53] presented zero-crossing theory based algorithms. Sobel [51] proposed the famous Sobel operator to compute the gradient map of an image, and then yielded edges by thresholding the gradient map. An extended version of Sobel, named Canny [6], added Gaussian smoothing as a preprocessing step and used the bi-threshold to get edges. In this way, Canny is more robust to noise. In fact, it is still very popular across various tasks now because of its notable efficiency. However, these early methods seem to have poor accuracy and thus are difficult to adapt to today's applications.

Later, researchers tended to manually design features using low-level cues such as intensity, gradient, and texture, and then employ sophisticated learning paradigm to classify edge and non-edge pixels [13, 44]. Konishi *et al.* [30] proposed the first data-driven methods by learning the probability distributions of responses that correspond to two sets of edge filters. Martin *et al.* [40] formulated changes in brightness, color, and texture as Pb features, and trained a classifier to combine the information from these features. Arbeláez *et al.* [2] developed Pb into gPb by using standard Normalized Cuts [48] to combine above local cues into a globalization framework. Lim [36] proposed novel features, Sketch tokens that can be used to represent the mid-level information. Dollár *et al.* [14] employed random decision forests to represent the structure presented in local image patches. Inputting color and gradient features, the structured forests output high-quality edges. However, all the above methods are developed based on handcrafted features, which has limited ability to represent high level information for semantically meaningful edge detection.

With the vigorous development of deep learning recently, a series of deep learning based approaches have been invented. Ganin *et al.* [19] proposed N^4-Fields that combines CNNs with the nearest neighbor search. Shen *et al.* [47] partitioned contour data into subclasses and fit each subclass by learning model parameters. Hwang *et al.* [26] considered contour detection as a per-pixel classification problem. They employed DenseNet [27] to extract a feature vector for each pixel, and then SVM classier was used to classify each pixel into the edge or non-edge class. Xie *et al.* [58] recently developed an efficient and accurate edge detector, HED, which performs image-to-image training and prediction. This holistically-nested architecture connects their side output layers, which is composed of one *conv* layer with kernel size 1, one *deconv* layer and one softmax layer, to the last *conv* layer of each stage in VGG16 [50]. More recently, Liu *et al.* [37] used relaxed label generated by bottom-up edges to guide the training process of HED, and achieved some improvement. Li *et al.* [35] proposed a complex model for unsupervised learning of edge detection, but the performance is worse than training on the limited BSDS500 dataset.

The aforementioned CNN-based models have advanced the state-of-the-art significantly, but all of them lost some useful hierarchical CNN features when classifying pixels to edge or non-edge class. These methods usually only adopt CNN features from the last layer of each *conv* stage. To fix this case, we propose a fully convolutional network to combine features from each CNN layer efficiently. We will detail our method below.

3. Richer Convolutional Features (RCF)

3.1. Network Architecture

Inspired by previous literature in deep learning [20, 38, 43, 58], we design our network by modifying VGG16 network [50]. VGG16 network that composes of 13 *conv* layers and 3 fully connected layers has achieved state-of-the-art in a variety of tasks, such as image classification [50] , object detection [20,21,43] and *etc*. Its *conv* layers are divided into five stages, in which a pooling layer is connected after each stage. The useful information captured by each *conv* layer becomes coarser with its receptive field size increasing. Detailed receptive field sizes of different layers can be seen in Tab. 1. The use of this rich hierarchical information is hypothesized to help a lot. The starting point of our network design lies here.

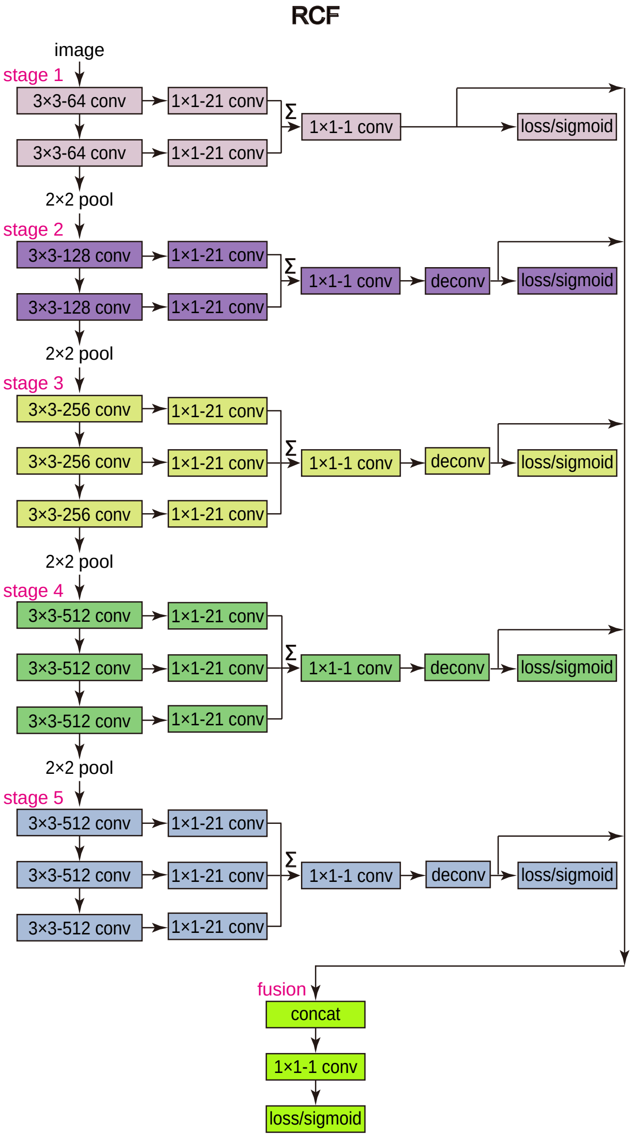

Figure 2: Our RCF network architecture. The input is an image with arbitrary sizes, and our network outputs an edge possibility map in the same size.

Table 1: Detailed receptive field and stride sizes of standard VGG16 net [50].

layer	conv1_1	conv1_2	pool1	conv2_1	conv2_2	pool2
rf size	3	5	6	10	14	16
stride	1	1	2	2	2	4
layer	conv3_1	conv3_2	conv3_3	pool3	conv4_1	conv4_2
rf size	24	32	40	44	60	76
stride	4	4	4	8	8	8
layer	conv4_3	pool4	conv5_1	conv5_2	conv5_3	pool5
rf size	92	100	132	164	196	212
stride	8	16	16	16	16	32

The novel network proposed by us is shown in Fig. 2. Compared with VGG16, our modifications can be described as following:

- We cut all the fully connected layers and the *pool5* layer. On the one side, we remove the fully connected layers due to the fact that they do not align with our design of fully convolutional network. On the other hand, adding *pool5* layer will increase the stride by two times, and it's harmful for edge localization.

- Each *conv* layer in VGG16 is connected to a *conv* layer with kernel size 1×1 and channel depth 21. And the resulting layers in each stage are accumulated using an *eltwise* layer to attain hybrid features.

- An $1 \times 1 - 1$ *conv* layer follows each *eltwise* layer. Then, a *deconv* layer is used to up-sample this feature map.

- A *cross-entropy loss* / *sigmoid* layer is connected to the up-sampling layer in each stage.

- All the up-sampling layers are concatenated. Then an 1×1 *conv* layer is used to fuse feature maps from each stage. At last, a *cross-entropy loss* / *sigmoid* layer is followed to get the fusion loss / output.

Hence, we combine hierarchical features from all the *conv* layers into a holistic framework, in which all of the parameters are learned automatically. Since receptive field sizes of *conv* layers in VGG16 are different from each other, our network can learn multiscale, including low-level and object-level, information that is helpful to edge detection. We show

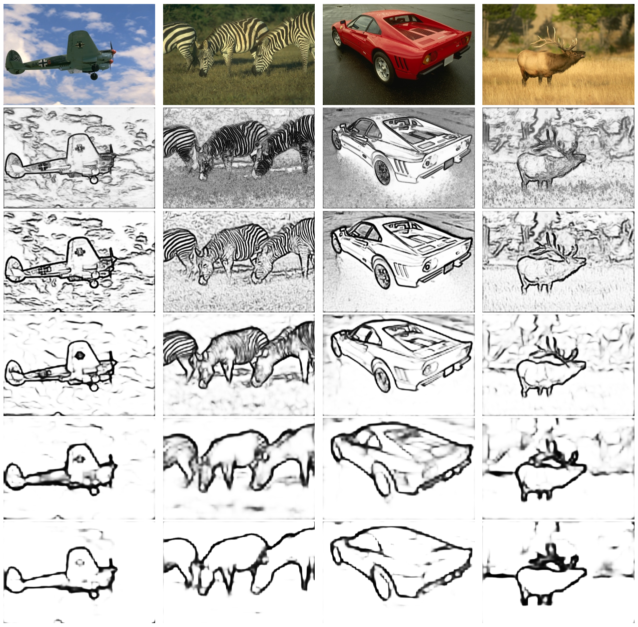

Figure 3: Several examples of the outputs in each stage of RCF. The top line is the original images from BSDS500 [2]. From second to sixth line is the output of stage 1, 2, 3, 4 and 5 respectively.

the intermediate results from each stage in Fig. 3. From top to bottom, the edge response becomes coarser while obtaining strong response at the larger object or object part boundaries. It is consistent with our expectation, in which *conv* layers will learn to detect the larger objects with the receptive field size increasing. Since our RCF model combines all the accessible *conv* layers to employ richer features, it is expected to achieve a boost in accuracy.

3.2. Annotator-robust Loss Function

Edge datasets in this community are usually labeled by several annotators using their knowledge about the presences of objects and object parts. Though humans vary in cognition, these human-labeled edges for the same image share high consistency. For each image, we average all the ground truth to generate an edge probability map, which ranges from 0 to 1. Here, 0 means no annotator labeled at this pixel, and 1 means all annotators have labeled at this pixel. We consider the pixels with edge probability higher than η as positive samples and the pixels with edge probability equal to 0 as negative samples. Otherwise, if a pixel is marked by fewer than η of the annotators, this pixel may be semantically controversial to be an edge point. Thus, whether regarding it as positive or negative samples may confuse networks. So we ignore pixels in this category.

We compute the loss at every pixel with respect to pixel label as

$$l(X_i;W) = \begin{cases} \alpha \cdot log\left(1 - P(X_i;W)\right) & if\ y_i = 0 \\ 0 & if\ 0 < y_i \le \eta \\ \beta \cdot log\ P(X_i;W) & otherwise, \end{cases} \tag{1}$$

in which

$$\begin{aligned} \alpha &= \lambda \cdot \frac{|Y^+|}{|Y^+| + |Y^-|} \\ \beta &= \frac{|Y^-|}{|Y^+| + |Y^-|}. \end{aligned} \tag{2}$$

Y^+ and Y^- denote positive sample set and negative sample set respectively. The hyper-parameter λ is to balance positive and negative samples. The activation value (CNN feature vector) and ground truth edge probability at pixel i are presented by X_i and y_i, respectively. $P(X)$ is the standard sigmoid function, and W denotes all the parameters that will be learned in our architecture. Therefore, our improved loss function can be formulated as

$$L(W) = \sum_{i=1}^{|I|} \Big(\sum_{k=1}^{K} l(X_i^{(k)};W) + l(X_i^{fuse};W) \Big), \tag{3}$$

where $X_i^{(k)}$ is the activation value from stage k while X_i^{fuse} is from fusion layer. $|I|$ is the number of pixels in image I, and K is the number of stages (equals to 5 here).

3.3. Multiscale Hierarchical Edge Detection

In single scale edge detection, we input an original image into our fine-tuned RCF network, then, the output is an edge probability map. To further improve the quality of edges, we use image pyramids during testing. Specifically, we resize an image to construct an image pyramid, and each of these images is input to our single-scale detector separately. Then, all resulting edge probability maps are resized to original image size using bilinear interpolation. At last, these maps are averaged to get a final prediction map. Fig. 4 shows a visualized pipeline of our multiscale algorithm. We also try to use weighted sum, but we find the simple average works very well. Considering the trade-off between accuracy and speed, we use three scales 0.5, 1.0, and 1.5 in this paper. When evaluating on BSDS500 [2] dataset, this simple multiscale strategy improves the ODS F-measure from 0.806 to 0.811, though the speed drops from 30 FPS to 8 FPS. See Sec. 4 for details.

3.4. Comparison With HED

The most obvious difference between our RCF and HED [58] is in three parts. First, HED only considers the last *conv* layer in each stage of VGG16, in which lots of helpful information to edge detection is missed. In contrast to it, RCF

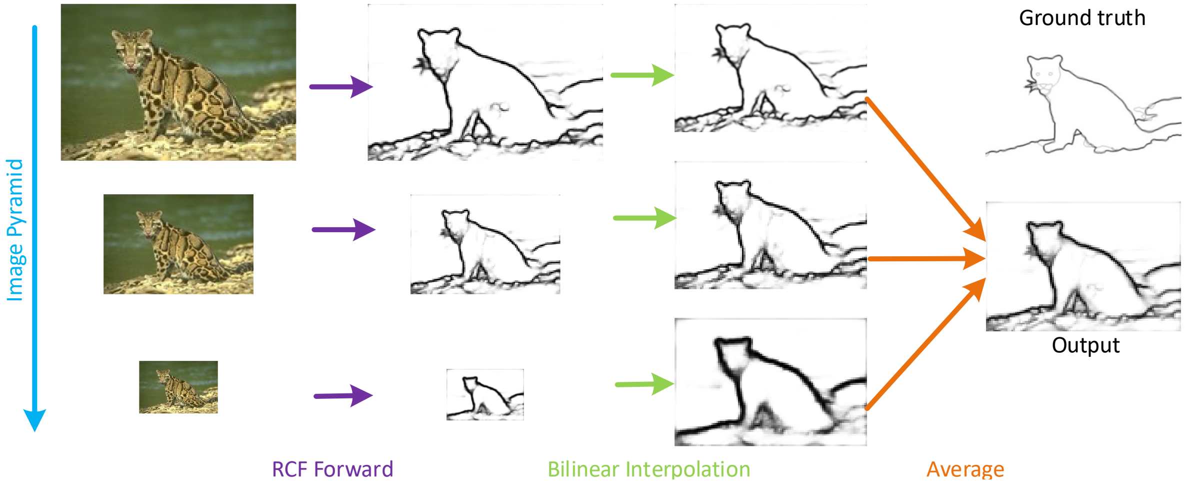

Figure 4: The pipeline of our multiscale algorithm. The original image is resized to construct an image pyramid. And these multiscale images are input to RCF network for a forward pass. Then, we use bilinear interpolation to restore resulting edge respondence maps to original sizes. A simple average of these edge maps will output high-quality edges.

uses richer features from all the *conv* layers, thus it can capture more object or object part boundaries accurately across a larger range of scales. Second, a novel loss function is proposed to treat training examples properly. We only consider the edge pixels that most annotators labeled as positive samples, since these edges are highly consistent and thus easy to train. Besides, we ignore edge pixels that are marked by a few annotators because of their confusing attributes. Thirdly, we use multiscale hierarchy to enhance edges. Our evaluation results demonstrate the strengths (2.3% improvement in ODS F-measure over HED) of these choices. See Sec. 4 for details.

4. Experiments

We implement our network using the publicly available Caffe [28] which is well-known in this community. The VGG16 model that is pre-trained on ImageNet [11] is used to initialize our network. We change the stride of pool4 layer to 1 and use the atrous algorithm to fill the holes. In RCF training, the weights of 1×1 *conv* layer in stage 1-5 are initialized from zero-mean Gaussian distributions with standard deviation 0.01 and the biases are initialized to 0. The weights of 1×1 *conv* layer in fusion stage are initialized to 0.2 and the biases are initialized to 0. Stochastic gradient descent (SGD) minibatch samples 10 images randomly in each iteration. For other SGD hyper-parameters, the global learning rate is set to 1e-6 and will be divided by 10 after every 10k iterations. The momentum and weight decay are set to 0.9 and 0.0002 respectively. We run SGD for 40k iterations totally. The parameters η and λ in loss function are also set depending on training data. All experiments in this paper are finished using a NVIDIA TITAN X GPU.

Given an edge probability map, a threshold is needed to produce the edge image. There are two choices to set this threshold. The first one is referred as optimal dataset scale (ODS) which employs a fixed threshold for all images in the dataset. And the second is called optimal image scale (OIS) which selects an optimal threshold for each image. We use F-measure ($\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$) of both ODS and OIS in our experiments.

4.1. BSDS500 Dataset

BSDS500 [2] is a widely used dataset in edge detection. It is composed of 200 training, 100 validation and 200 test images, and each image is labeled by 4 to 9 annotators. We utilize the training and validation sets for finetuning, and test set for evaluation. Data augmentation is the same as [58]. Inspired by the previous work [29, 37, 59], we mix augmentation data of BSDS500 with flipped PASCAL VOC Context dataset [42] as training data. When training, we set loss parameters η and λ to 0.5 and 1.1, respectively. When evaluating, standard non-maximum suppression (NMS) [14] is applied to thin detected edges. We compare our method with some non-deep-learning algorithms, including Canny [6], EGB [16], gPb-UCM [2], ISCRA [45], MCG [3], MShift [10], NCut [48], SE [14], and OEF [24], and some recent deep learning based approaches, including DeepContour [47], DeepEdge [4], HED [58], HFL [5], MIL+G-DSN+MS+NCuts [29] and *etc*.

Fig. 5 shows the evaluation results. The performance of human eye in edge detection is known as 0.803 ODS F-measure. Both single-scale and multiscale (MS) versions of RCF achieve better results than humans. When comparing with HED [58], ODS F-measures of our RCF-MS and RCF are 2.3% and 1.8% higher than it, respectively. And

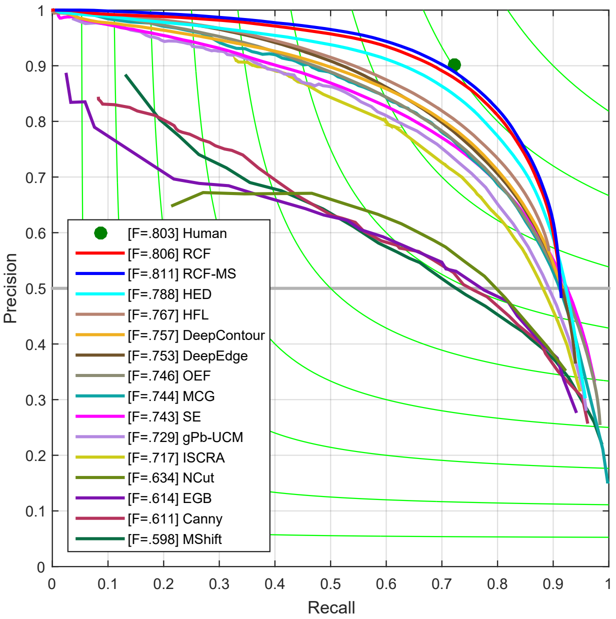

Figure 5: The evaluation results on standard BSDS500 [2] dataset. Both single-scale and multiscale versions of RCF achieve better performance than humans.

Table 2: The comparison with some competitors on BSDS500 [2] dataset. † means GPU time. The top three results are highlighted in red, green and blue.

Method	ODS	OIS	FPS
Canny [6]	.611	.676	**28**
EGB [16]	.614	.658	**10**
MShift [10]	.598	.645	1/5
gPb-UCM [2]	.729	.755	1/240
Sketch Tokens [36]	.727	.746	1
MCG [3]	.744	.777	1/18
SE [14]	.743	.763	2.5
OEF [24]	.746	.770	2/3
DeepContour [47]	.757	.776	1/30†
DeepEdge [4]	.753	.772	1/1000†
HFL [5]	.767	.788	5/6†
N^4-Fields [19]	.753	.769	1/6†
HED [58]	.788	.808	**30**†
RDS [37]	.792	.810	**30**†
CEDN [59]	.788	.804	**10**†
MIL+G-DSN+MS+NCuts [29]	**.813**	**.831**	1
RCF	**.806**	**.823**	**30**†
RCF-MS	**.811**	**.830**	8†

the precision-recall curves of our methods are also higher than HED's. These significant improvements demonstrate the effectiveness of our richer convolutional features. All the *conv* layers contain helpful hierarchical information, not only the last one in each convolution stage.

We show statistic comparison in Tab. 2. From RCF to RCF-MS, the ODS F-measure increases from 0.806 to 0.811, though the speed drops from 30 FPS to 8 FPS. It proves the validity of our multiscale strategy. We also observe an interesting phenomenon in which the RCF curves are not as long as other methods when evaluated using the default parameters in BSDS500 benchmark. It may suggest that RCF tends only to remain very confident edges. Our methods also achieve better results than recent edge detectors, such as RDS [37] and CEDN [59]. RDS uses relaxed laebls and extra training data to retrain the HED network, and it improves 0.4% of ODS F-measure compared with HED. In contrast, the F-measure of our RCF method is 1.4% higher in ODS F-measure than RDS. It demonstrates our improvement is not trivial or ad hoc.

We can see that RCF achieves the best tarde-off between effectiveness and efficiency. Although MIL+G-DSN+MS+NCuts [29] achieves a little better accuracy than our methods, our RCF and RCF-MS are much fastest than it. The single-scale RCF achieves **30** FPS, and RCF-MS can also achieve **8** FPS. Note that our RCF network only adds some 1×1 *conv* layers to HED, so the time consumption is almost same as HED. Besides, starting from HED, Iasonas *et al.* [29] added some useful components to it, such as Multiple Instance Learning (MIL) [12], G-DSN [32], multiscale, extern training data with PASCAL Context dataset [42] and Normalized Cuts [2]. Our proposed methods are much simpler than [2]. Since our edge detectors are simple and efficient, it is easy to apply them in various high-level vision tasks.

4.2. NYUD Dataset

NYUD [49] dataset is composed of 1449 densely labeled pairs of aligned RGB and depth images. Recently many works have conducted edge evaluation on it, such as [14, 57]. Gupta *et al.* [22] split NYUD dataset into 381 training, 414 validation and 654 testing images. We follow their settings and train our RCF network using training and validation sets in full resolution as in HED [58].

We utilize depth information by using HHA [23], in which depth information is encoded into three channels: horizontal disparity, height above ground, and angle with gravity. Thus HHA features can be represented as a color image. Then, two models for RGB images and HHA feature images are trained separately. We rotate the images and corresponding annotations to 4 different angles (0, 90, 180 and 270 degrees) and flip them at each angle. In the training process, λ is set to 1.2 for both RGB and HHA. Since NYUD only has one ground truth for each image, η is useless here. Other network settings are the same as used for BSDS500. At testing, the final edge predictions are defined

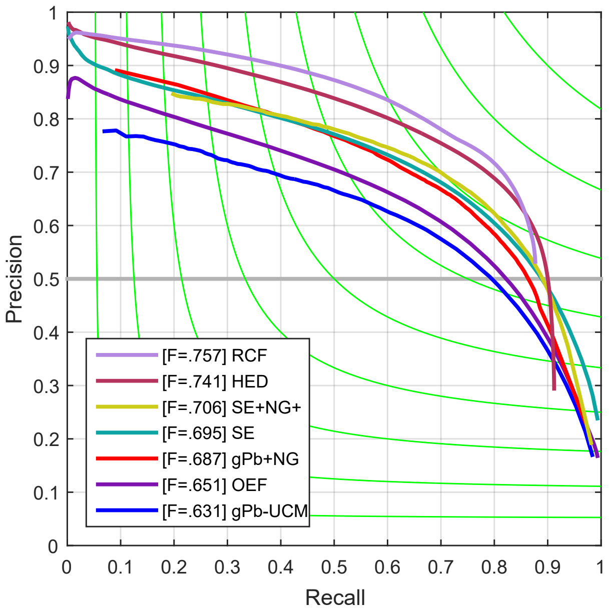

Figure 6: The evaluation results on NYUD [49] dataset. RCF is referred to single-scale version here.

by averaging the outputs of RGB model and HHA model. When evaluating, we increase localization tolerance, which controls the maximum allowed distance in matches between predicted edges and ground truth, from 0.0075 to 0.011, because images in NYUD dataset are larger than images in BSDS500 dataset.

Table 3: The comparison with some competitors on NYUD dataset [49]. †means GPU time.

Method	ODS	OIS	FPS
OEF [24]	.651	.667	1/2
gPb-UCM [2]	.631	.661	1/360
gPb+NG [22]	.687	.716	1/375
SE [14]	.695	.708	5
SE+NG+ [23]	.706	.734	1/15
HED-HHA [58]	.681	.695	**20**†
HED-RGB [58]	.717	.732	**20**†
HED-RGB-HHA [58]	.741	.757	10†
RCF-HHA	.705	.715	**20**†
RCF-RGB	.729	.742	**20**†
RCF-HHA-RGB	**.757**	**.771**	10†

We only compare our single-scale version of RCF with some famous competitors. OEF [24] and gPb-UCM [2] only use RGB images, while other methods employ both depth and RGB information. The precision-recall curves are shown in Fig. 6. RCF achieves the best performance on NYUD dataset, and the second place is HED [58]. Tab. 3 shows statistical comparison. We can see that RCF achieves better results than HED not only on separate HHA or RGB data, but also on the merged HHA-RGB data. For HHA and RGB data, ODS F-measure of RCF is 2.4% and 1.2% higher than HED, respectively. For merging HHA-RGB data, RCF is 1.6% higher than HED. Furthermore, HHA edges perform worse than RGB, but averaging HHA and RGB edges achieves much higher results. It suggests that combining different types of information is very useful for edge detection, and it may be the reason why OEF and gPb-UCM perform worse than other methods.

4.3. Multicue Dataset

Recently, Multicue dataset is proposed by Mély *et al.* [41] to study psychophysics theory for boundary detection. It is composed of short binocular video sequences of 100 challenging natural scenes captured by a stereo camera. Each scene contains a left and a right view short (10-frame) color sequences. The last frame of the left images for each scene is labeled for two annotations, object boundaries and low-level edges. Unlike people who usually use boundary and edge interchangeably, they strictly defined boundary and edge according to visual perception at different stages. Thus, boundaries are referred to the boundary pixels of meaningful objects, and edges are abrupt pixels at which the luminance, color or stereo change sharply. In this subsection, we use boundary and edge as defined by Mély *et al.* [41] while considering boundary and edge having the same meaning in previous sections.

As done in Mély *et al.* [41] and HED [58], we randomly split these human-labeled images into 80 training and 20 test samples, and average the scores of three independent trials as final results. When training on Multicue, λ is set to 1.1, and η is set to 0.4 for boundary task and 0.3 for edge task. For boundary detection task, we use learning rate 1e-6 and run SGD for 2k iterations. For edge detection task, we use learning rate 1e-7 and run SGD for 4k iterations. We augment the training data as we do on NYUD dataset. Since the image resolution of Multicue is very high, we randomly crop 500×500 patches from original images.

We show evaluation results in Tab. 4. Our proposed RCF achieve substantially higher results than HED. For boundary task, RCF-MS is 1.1% ODS F-measure higher and 1.4% OIS F-measure higher than HED. For edge task, RCF-MS is 0.9% ODS F-measure higher than HED. Note that the fluctuation of RCF is also smaller than HED, which suggests RCF is more robust over different kinds of images. Some qualitative results are shown in Fig. 7.

4.4. Network Discussion

To further explore the effectiveness of our network architecture, we implement some mixed networks using VGG16 [50] by connecting our richer feature side outputs to some

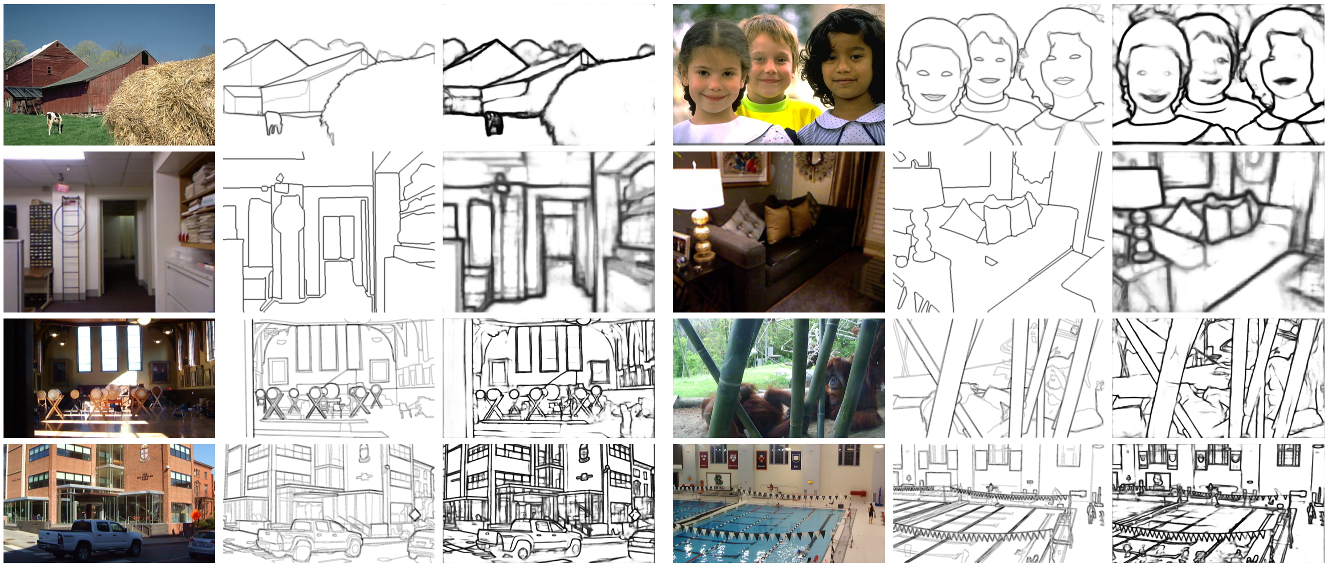

Figure 7: Some examples of RCF. From top to bottom: BSDS500 [2], NYUD [49], Multicue-Boundary [41], and Multicue-Edge [41]. From left to right: origin image, ground truth, RCF edge map, origin image, ground truth, and RCF edge map.

Table 4: The comparison with some competitors on Multicue dataset [41].

Method	ODS	OIS
Human-Boundary [41]	.760 (.017)	–
Multicue-Boundary [41]	.720 (.014)	–
HED-Boundary [58]	.814 (.011)	.822 (.008)
RCF-Boundary	.817 (**.004**)	.825 (**.005**)
RCF-MS-Boundary	**.825** (.008)	**.836** (.007)
Human-Edge [41]	.750 (.024)	–
Multicue-Edge [41]	.830 (**.002**)	–
HED-Edge [58]	.851 (.014)	**.864** (.011)
RCF-Edge	.857 (.004)	.862 (**.004**)
RCF-MS-Edge	**.860** (.005)	**.864** (**.004**)

convolution stages while connecting side outputs of HED to the other stages. With training only on BSDS500 [2] dataset and testing on the single scale, evaluation results of these mixed networks are shown in Tab. 5. The last two lines of this table correspond to HED and RCF, respectively. We can observe that all of these mixed networks perform better than HED and worse than RCF that is fully connected to RCF side outputs. It clearly demonstrates the importance of our strategy of richer convolutional features.

In order to investigate whether including additional nonlinearity helps, we connecting ReLU layer after $1 \times 1 - 21$ or $1 \times 1 - 1$ *conv* layers in each stage. However, the network performs worse. Especially, when we attempt to add nonlinear layers to $1 \times 1 - 1$ *conv* layers, the network can not converge properly.

Table 5: Results of some thought networks.

RCF Stage	HED Stage	ODS	OIS
1, 2	3, 4, 5	.792	.810
2, 4	1, 3, 5	.795	.812
4, 5	1, 2, 3	.790	.810
1, 3, 5	2, 4	.794	.810
3, 4, 5	1, 2	.796	.812
–	1, 2, 3, 4, 5	.788	.808
1, 2, 3, 4, 5	–	.798	.815

5. Conclusion

In this paper, we propose a novel CNN architecture, RCF, that makes full use of semantic and fine detail features to carry out edge detection. We carefully design it as an extensible architecture. The resulting RCF method can produce high-quality edges very efficiently, and this makes it promising to be applied in other vision tasks. RCF architecture can be seen as a development direction of fully connected network, like FCN [38] and HED [58]. It would be interesting to explore the usefulness of our network architecture in other hot topics, such as salient object detection and semantic segmentation. Source code is available at https://github.com/yun-liu/rcf.

Acknowledgments We would like to thank the anonymous reviewers for their useful feedbacks. This research was supported by NSFC (NO. 61572264, 61620106008), Huawei Innovation Research Program (HIRP), and CAST young talents plan.

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *IEEE CVPR*, pages 2294–2301. IEEE, 2009. 1

[2] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 33(5):898–916, 2011. 1, 2, 4, 5, 6, 7, 8

[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE CVPR*, pages 328–335, 2014. 1, 5, 6

[4] G. Bertasius, J. Shi, and L. Torresani. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE CVPR*, pages 4380–4389, 2015. 1, 5, 6

[5] G. Bertasius, J. Shi, and L. Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *IEEE ICCV*, pages 504–512, 2015. 5, 6

[6] J. Canny. A computational approach to edge detection. *IEEE TPAMI*, (6):679–698, 1986. 2, 5, 6

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 1

[8] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, and Z. Tu. HFS: Hierarchical feature selection for efficient image segmentation. In *ECCV*, pages 867–882. Springer, 2016. 1

[9] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *IEEE CVPR*, 2014. 1

[10] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002. 5, 6

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, pages 248–255. IEEE, 2009. 5

[12] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71, 1997. 6

[13] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *IEEE CVPR*, volume 2, pages 1964–1971. IEEE, 2006. 2

[14] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE TPAMI*, 37(8):1558–1570, 2015. 1, 2, 5, 6, 7

[15] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973. 2

[16] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 5, 6

[17] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE TPAMI*, 30(1):36–51, 2008. 1

[18] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE TOC*, 100(6):616–628, 1975. 2

[19] Y. Ganin and V. Lempitsky. N^4-Fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, pages 536–551. Springer, 2014. 1, 2, 6

[20] R. Girshick. Fast R-CNN. In *IEEE ICCV*, pages 1440–1448, 2015. 1, 3

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE CVPR*, pages 580–587, 2014. 1, 3

[22] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *IEEE CVPR*, pages 564–571, 2013. 6, 7

[23] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, pages 345–360. Springer, 2014. 6, 7

[24] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *IEEE CVPR*, pages 1732–1740, 2015. 5, 6, 7

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE CVPR*, pages 770–778, 2016. 2

[26] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015. 1, 2

[27] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014. 2

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678. ACM, 2014. 5

[29] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386*, 2015. 5, 6

[30] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE TPAMI*, 25(1):57–74, 2003. 2

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[32] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 5, 2015. 6

[33] M. Leordeanu, R. Sukthankar, and C. Sminchisescu. Generalized boundaries from multiple image interpretations. *IEEE TPAMI*, 36(7):1312–1324, 2014. 1

[34] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, pages 379–387, 2016. 1

[35] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. In *IEEE CVPR*, pages 1619–1627, 2016. 2

[36] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE CVPR*, pages 3158–3165, 2013. 1, 2, 6

[37] Y. Liu and M. S. Lew. Learning relaxed deep supervision for better edge detection. In *IEEE CVPR*, pages 231–240, 2016. 2, 5, 6

[38] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, pages 3431–3440, 2015. 1, 3, 8

[39] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, 207(1167):187–217, 1980. 2

[40] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE TPAMI*, 26(5):530–549, 2004. 1, 2

[41] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre. A systematic comparison between visual cues for boundary detection. *Vision research*, 120:93–107, 2016. 7, 8

[42] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE CVPR*, pages 891–898, 2014. 5, 6

[43] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 1, 3

[44] X. Ren. Multi-scale improves boundary detection in natural images. In *ECCV*, pages 533–545. Springer, 2008. 2

[45] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *IEEE CVPR*, pages 2011–2018, 2013. 5

[46] G. S. Robinson. Color edge detection. *Optical Engineering*, 16(5):165479–165479, 1977. 2

[47] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE CVPR*, pages 3982–3991, 2015. 2, 5, 6

[48] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, 2000. 2, 5

[49] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 6, 7, 8

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 3, 7

[51] I. Sobel. Camera models and machine perception. Technical report, DTIC Document, 1970. 2

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE CVPR*, pages 1–9, 2015. 1

[53] V. Torre and T. A. Poggio. On edge detection. *IEEE TPAMI*, (2):147–163, 1986. 2

[54] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013. 1

[55] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE TPAMI*, 13(10):992–1006, 1991. 1

[56] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, J. Feng, Y. Zhao, and S. Yan. Stc: A simple to complex framework for weakly-supervised semantic segmentation. *IEEE TPAMI*, 2016. 1

[57] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, pages 584–592, 2012. 1, 6

[58] S. Xie and Z. Tu. Holistically-nested edge detection. In *IJCV*. Springer, 2017. 2, 3, 4, 5, 6, 7, 8

[59] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang. Object contour detection with a fully convolutional encoder-decoder network. *arXiv preprint arXiv:1603.04530*, 2016. 5, 6

[60] Z. Zhang, Y. Liu, T. Bolukbasi, M.-M. Cheng, and V. Saligrama. Bing++: A fast high quality object proposal generator at 100fps. *arXiv preprint arXiv:1511.04511*, 2015. 1

[61] S. Zheng, V. A. Prisacariu, M. Averkiou, M.-M. Cheng, N. J. Mitra, J. Shotton, P. H. Torr, and C. Rother. Object proposals estimation in depth image using compact 3d shape manifolds. In *German Conference on Pattern Recognition*, pages 196–208, 2015. 1

[62] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405. Springer, 2014. 1