Learning Hybrid Convolutional Features for Edge Detection

Xiaowei Hu, Yun Liu, Kai Wang, Bo Ren*

CCCE, Nankai University, Tianjin, P.R.China, 300350

Abstract

We present a novel convolutional neural network (CNN) based pipeline which can effectively fuse multi-level information extracted from different intermediate layers generating hybrid convolutional features (HCF) for edge detection. Different from previous methods, the proposed method fuses multi-level information in a feature-map based manner. The produced hybrid convolutional features can be used to perform high-quality edge detection. The edge detector is also computationally efficient, because it detects edges in an image-to-image way without any post-processing. We evaluate the proposed method on three widely used datasets for edge detection including BSDS500, NYUD and Multicue, and also test the method on Pascal VOC'12 dataset for object contour detection. The results show that HCF achieves an improvement in performance over the state-of-the-art methods on all four datasets. On BSDS500 dataset, the efficient version of the proposed approach achieves ODS F-score of **0.804** with a speed of **22 fps** and the high-accuracy version achieves ODS F-score of **0.814** with **11 fps**.

Keywords: Edge detection; Hybrid convolutional features; Feature integration; High accuracy; Human performance

1. Introduction

In this paper, we focus on the classical problem of detecting edges and boundaries in natural images. As a fundamental problem, edge detection is greatly important to a lot of high-level computer vision tasks, such as object detection [1, 2], biological image processing [3, 4, 5, 6, 7], and image segmentation [8, 9, 10] etc. We propose an efficient and high-accuracy edge detector, which detects edges in an image-to-image way without any post-processing.

Edge detection has an extremely rich history. Early detectors are mainly based on local cues of brightness, colors and gradients, such as Sobel [11], Canny [12], etc. However, local cues are too simple to capture high-level information and it has been proved that high-level information is important to edge detection. Traditional learning-based methods [13, 14, 15, 16] utilize carefully designed manual features to extract rich features, and employ supervised learning algorithms to classify edge/non-edge pixels. However, these methods are still not sufficient to recover semantic information subjecting to complex image contents [17].

In recent years, CNNs have made significant progress on many computer vision tasks, such as image recognition [18, 19], object detection [1, 2], semantic segmentation [20, 21], etc. It demonstrates that CNNs have the capacity for automatically extracting high-quality semantic-level features. When referring to edge detection, a series of CNN-based edge detectors [17, 22, 23, 24, 25, 26, 27] have been proposed. Some early CNN-based methods utilize the last fully-connected layer as the features to classify edge/non-edge pixels; recent methods fuse multi-level information for detect edges.

To fuse multi-level information, there are two typical modern CNN architectures, fully convolutional network (FCN) [20] and holistically-nested network (HED) [24]. FCN is originally designed for semantic segmentation and also achieves high performance on edge detection as mentioned in [24]. This method first produces a probability map from the last layer of CNNs and then updates the probability map by adding it with probability maps produced from lower-level features level by level. HED is a much more effective architecture than FCN for edge detection. It combines multiple probability maps produced in a parallel way to fuse multi-level information. Both FCN and HED fuse multi-level information in a probability map based fashion. These probability-map based methods [24, 25, 27] firstly produce several probability maps from different single-level features and then combine these probability maps to fuse multi-level information, so each probability map is only related to a single-level feature and independent of each other. Thus different level features cannot directly interact with each other in these methods.

Therefore, we develop a novel CNN-based pipeline which fuses multi-level information in a feature-map based fashion instead of the probability-map based fashion. Specifically, the proposed feature-map based method combines multi-level features, generating hybrid convolutional features (HCF) before predicting one probability map. Hybrid convolutional features serve as a layer that the layer after it can explore the feature interactions among hybrid convolutional features. Thus this method can extract much richer features and these features can directly interact with each other among different levels. Inspired by recent research showing that adding auxiliary losses

*Corresponding author
Email address: rb@nankai.edu.cn (Bo Ren)

can make the learning process transparent, we propose the architecture with a master branch and an auxiliary branch. The master branch fuses both high- and low- level information to detect high-accuracy edges; the auxiliary branch only fuses high-level information to help the master branch to extract powerful high-level features.

We evaluate the proposed method on three widely used datasets for edge detection including BSDS500 [13], NYUD [28] and Multicue [29]. Different from edge detection, object contour detection [26] is intended to detect the contours of objects which belong to several interesting classes, and we also test the proposed method on Pascal VOC'12 [30] dataset for object contour detection. The results demonstrate that the proposed HCF obtains the state-of-the-art performance over all four datasets with a fast speed. On BSDS500 dataset, the efficient version of the proposed approach achieves ODS F-score of **0.804** with a speed of **22 fps** and the high-accuracy version achieves ODS F-score of **0.814** with **11 fps**.

Our key contributions are summarized below:

- We present a novel CNN-based pipeline which fuses multi-level information in a feature-map based fashion and produces high-quality hybrid convolutional features (HCF) for detecting edges in an image-to-image way without any post-precessing. Meanwhile, we add an auxiliary branch to help the master branch extract powerful high-level features.

- We further explore the effectiveness of different fundamental CNNs including VGGNet and ResNet, and combine VGGNet and ResNet together as HybridNet for the proposed method.

2. Related work

Traditional edge detectors. Early edge detection methods are mainly based on local filters, such as Sobel [11], Canny [12], etc. Sobel computes the gradient map of an image and then obtains edges by thresholding the gradient map. Further improved from Sobel, Canny adds Gaussian smoothing as a preprocessing step and applies double threshold to generate edges. Konishi *et al.* [14] presented the first data-driven method, Statistical Edge. The authors formulated edge detection as statistical inference and learned the probability distributions of filter responses to classify pixels into edge/non-edge. With the advance of manually designed features, Martin *et al.* [31] used different types of local cues in their probabilistic method. Arbelaez *et al.* [13] developed an improved method which utilizes a globalization technique to extract global information to boost the performance of edge detection. Dollár *et al.* [16] employed random decision forests to learn different local structures clustered by K-means for fast edge detection.

CNN-based edge detectors. Recently deep convolutional networks [18, 19, 32, 33] have shown significantly improved performance on image classification and object detection. Krizhevsky *et al.* presented a CNN architecture named AlexNet (8 layers), which starts the applications of deep CNNs on computer vision tasks. Compared with AlexNet, VGGNet (16 layers) and GoogLeNet (22 layers) have a deeper and more complex architecture and are more powerful on computer vision tasks. The extremely deep ResNet (50, 101, 152 layers) has shown its high capacity on image recognition and object detection. In this work, we use both VGGNet (16 layers) and ResNet (50 layers) as the fundamental CNNs of the proposed method to perform experiments for edge detection.

The latest wave of edge detectors gets benefits from deep CNNs, which can automatically extract multi-level features from images. Ganin *et al.* [17] proposed the first CNN-based edge detector, N^4-Fields, which employs CNNs to extract features of image patches and uses the nearest neighbor to match extracted features to predefined features of edge patches. Shen *et al.* [22] regarded edge detection as a classification problem and classified each patch of images into one of several predefined shape classes. Bertasius *et al.* [23] presented a cascaded edge detector which first uses structured edge (SE) [16] to generate candidate contour points and employs a CNN to produce a final boundary map. These early CNN-based methods mostly predict edge maps in a patch-to-class paradigm. Xie and Tu [24] proposed a novel approach, holistically-nested network (HED), detecting edges in an image-to-image way. HED is a multiscale network and combines multiple side outputs to fuse multi-level information and predicts edge maps with a fast speed. Yang *et al.* [26] extended edge detection to object contour detection, which aims to detect higher-level object contours. They also presented a novel architecture which forward passes an encoder-decoder network to detect object contours. Maninis *et al.* [27] proposed a HED-style edge detector based on ResNet, for predicting both the intensity and orientation of contours. This is also a probability-map based method and gets benefit from post-processing for edge detection.

3. The proposed method

In this section, we introduce the proposed fully convolutional architecture, namely hybrid convolutional features (HCF), to perform edge detection in an image-to-image way. In Section 3.1, we will introduce our network architecture of HCF in detail. In Section 3.2, we formally introduce how to train HCF in an end-to-end way. Finally, in Section 3.3, the designed HybridNet, which combines VGGNet and ResNet, is introduced.

3.1. Network Architecture

This section is devoted to introducing the proposed network architecture designed for fusing multi-level information in a feature-map based fashion. Our feature-map based method combines multi-level features before predicting probability maps and different level features can interact with each other in our method. Thus our method can more effectively integrate multi-level features than probability-map based methods.

To introduce our network architecture, we start from fundamental CNNs pre-trained on the ILSVRC CLS-LOC dataset,

such as VGGNet [18] and ResNet [19]. In the remainder of this section, we use VGGNet as the fundamental CNN to describe our network architecture. VGGNet has 13 convolutional layers which are divided into 5 stages. A convolution stage is defined by all convolutional layers between two adjacent pooling layers. In one stage, all convolutional layers have the same channels. For example, each convolutional layer in the 1st stage has 64 channels.

We utilize a 3×3 convolutional layer to extract a feature map with 16 channels from the last convolutional layer of each stage. Five feature maps will be extracted from the *conv1_2*, *conv2_2*, *conv3_3*, *conv4_3* and *conv5_3* layers. *conv**n**_**m*** means the **m** layer of **n** stage in VGGNet. These feature maps capture different level information from original images. These feature maps extracted from different stages of VGGNet generally have extremely different L^2 norm. From bottom to top of VGGNet, the L^2 norm of feature maps becomes smaller and smaller. Thus naively stacking feature maps leads to poor performance as the "larger" features dominate the "smaller" ones. In order to handle this problem, we apply L^2 normalization technique [34], to scale L^2 norm at each location in the feature maps to a value γ which is initalized to 10 in all experiments and use backpropagation to update the scale γ during training as described in [34].

As shown in Figure 1, in the master branch, we upsample all the feature maps to the size of original image and stack them together as hybrid convolutional features. Hybrid convolutional features serve as a layer that two convolutional layers after it can explore the feature interactions among hybrid convolutional features. Thus the method can more effectively integrates multi-level features. We set the kernel size of the first convolutional layer to 9×9 which can enlarge the receptive field of filters to incorporate larger context. Because the second convolutional layer's output is edge probability map, the layer is with an 1×1 kernel size. The output depths of the two layers are set to 32 and 1 respectively. Additionally, we also add an auxiliary branch that helps to capture high-level semantic information. A reasonable order of the learning process is to detect edges on a high-to-low strategy, i.e. extracting general shape of object first and then gradually confining to finer details. In this sense, high-level features are more important in the whole process since they lie at the basis of the detection. Thus, we emphasize the high-level features by using an auxiliary branch. Specifically, in the auxiliary branch, the architecture is similar with the master branch, but this branch only stacks high-level feature maps which are extracted from *conv3_3*, *conv4_3* and *conv5_3*, and independently produce an auxiliary edge map. Thus it can help the master branch to extract powerful high-level features. Except the last 1×1 layers followed by a sigmoid function to compute edge maps, because the output depths of rest additional layers are small and activation functions make features too sparse to capture enough information, these layers are not followed by any activation function.

In summary, our method fuses multi-level information in a feature based fashion. In the network, L^2 normalization technique is applied to deal with the problem of different scales of different feature maps and an auxiliary branch is added to help the master branch to extract powerful high-level features.

3.2. Formulation of HCF

In this section, we give the formulation of HCF and discuss the techniques used in training phase, including class balancing and an auxiliary loss. Using the notation as in [24], we denote $S = \{(X_n, Y_n), n = 1, ..., N\}$ as the training set, with X_n being the input image and $Y_n = \{y_j^{(n)}, j = 1, ...|X_n|\}$, $y_j^{(n)} \in \{0, 1\}$ being the pixelwise ground true labels. The sampled pair from the training set is denoted by (X, Y).

Class-balanced cross-entropy loss. To deal with the class imbalance problem in edge detection, we apply a class-balancing technique as described in [24]. A class-balancing scale β is utilized to balance the losses of two classes, edge and non-edge. We can define the class-balanced cross-entropy loss as:

$$\begin{aligned} \ell(W, w) = &-\beta \sum_{j \in Y_+} logPr(y_j = 1|X; W, w) \\ &-(1-\beta) \sum_{j \in Y_-} logPr(y_j = 0|X; W, w), \end{aligned} \tag{1}$$

where $\beta = |Y_-|/|Y|$ and $1 - \beta = |Y_+|/|Y|$. $|Y_-|$ and $|Y_+|$ mean the number of pixels of edge and non-edge respectively. Given the input image X, W means the weight of fundamental CNN such as VGGNet or ResNet and w means the newly added weight in the proposed method, $Pr(y_j = 1|X; W, w)$ is the probability of predicting $y_j = 1$, and the probability is obtained by applying a standard sigmoid activation function $\delta(\cdot)$ to the output of deep convolutional networks.

Auxiliary loss. We add an auxiliary branch to our architecture and this branch can help the master branch to extract powerful high-level features. We denote the parameters unique to the master branch as w_m, the parameters unique to the auxiliary branch as w_a and the rest shared parameters as W. Putting the master loss and the auxiliary loss together, we minimize the following objective function via standard stochastic gradient descent:

$$\mathcal{L}(W, w_m, w_a) = \ell_m(W, w_m) + \alpha \cdot \ell_a(W, w_a)) \tag{2}$$

where ℓ_m and ℓ_a means the losses of the master branch and the auxiliary branch respectively. We set the weight α of the auxiliary loss to different scales on different datasets. More details about the weight α can be found in Section 4.

3.3. The architecture of fundamental CNNs

Fundamental CNNs pretrained on the ILSVRC CLS-LOC dataset have much effect on the performance of CNN-based edge detectors. There are four fundamental CNNs, including VGGNet [18], GoogLeNet [33], ResNet [19]and DenseNet [35] which are widely used in computer vision. VGGNet and GoogLeNet have 16 layers and 22 layers respectively and they have comparable performance on the ILSVRC CLS-LOC dataset. Since VGGNet has a simple and multistage architecture, it is widely used in edge detection [23, 22, 24, 26]. ResNet is an extremely deep architecture, and achieves much better performance than VGGNet on the ILSVRC CLS-LOC dataset. DenseNet [35] is a more complex architecture which

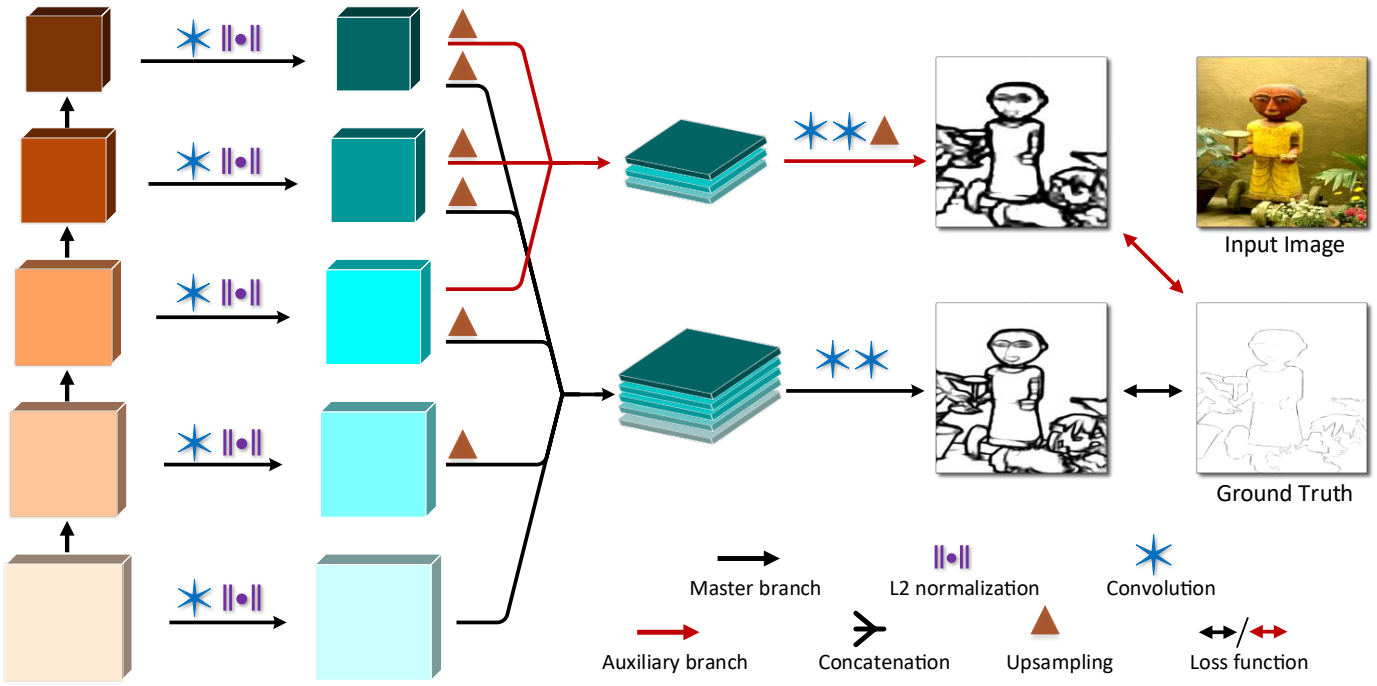

Figure 1: **The architecture of HCF based on VGGNet** (best viewed in color). The left convolutional network is VGGNet and from bottom to top the 5 layers (squares in the figure) mean *conv1_2*, *conv2_2*, *conv3_3*, *conv4_3* and *conv5_3* respectively. ***conv**n_m* means the **m** layer of **n** stage in VGGNet. The first step is using VGGNet to extracting different level features; the second step is combining the features and generating hybrid convolutional features (HCF); the last step is producing edge probability map from hybrid convolutional features. More details can be found in Section 3.1.

directly connects to each former layer within a dense block, so it is slower than VGGNet and ResNet on image classification. Because ResNet is much deeper than VGGNet and GoogLeNet, we use ResNet to extract high level features. In addition VGGNet and GoogLeNet have similar layers (16 layers vs 22 layers) and VGGNet is more simple and widely used than GoogLeNet, so we use VGGNet to extract low level features. Thus we use VGGNet and ResNet as two of the fundamental CNNs. When ResNet is used as the fundamental CNN, the master branch of HCF is built on *conv1*, *res2c*, *res3d*, *res4f* and *res5c* layers of ResNet; the auxiliary branch is built on the feature maps extracted from *res3d*, *res4f* and *res5c* layers. The remaining network is connected in a way similar to VGGNet described in Section 3.1.

Although ResNet achieves much better performance than VGGNet on the ILSVRC CLS-LOC dataset, the performance of edge detection based on ResNet has a limited improvement than VGGNet. The reason is that ResNet cannot extract high-quality low-level features. To handle this problem, we combine low-level layers of VGGNet and high-level layers of ResNet, including *conv1_2*, *conv2_2* and *conv3_3* of VGGNet, and *res3d*, *res4f* and *res5c* of ResNet, as the base layers of master branch of HCF, and only use high-level layers of ResNet as the base layers of the auxiliary branch. We name the hybrid convolutional network as HybridNet, which gets benefits from both ResNet and VGGNet. Particularly, we do some control experiments in Section 4.2.1 to evaluate the performance of the two fundamental CNNs including VGGNet and ResNet, and one HybridNet, which combines VGGNet and ResNet together.

3.4. Implement details

We implement our method using the publicly available *Caffe* [36], which is widely used in computer vision research, and build on the version released by the authors of DeepLab[1] [21]. During training, we set hyper-parameters according to the results on the validation set of BSDS500 dataset. The mini-batch size of stochastic gradient descent (SGD) is set to 10. The learning rate of SGD is initialized to 1e-6 and divided by 10 after 8k iterations. We set the momentum and weight decay to 0.9 and 0.0005 respectively and train the network with 10k iterations. The scale γ of L^2 normalization is initialized to 10 and the weight α of the auxiliary loss is finetuned on different datasets. Additional convolutional layers in our architecture are initialized by Xavier algorithm [37], except the last convolutional layer of each branch which is initialized to 0. All experiments in this work are carried out using a single NVIDIA TITAN X GPU.

4. Experiments

In this section, we first do control experiments to assess each component of our method, including hybrid convolutional features, the auxiliary loss and HybridNet. Then we compare our method with competitive methods on four benchmarks, including BSDS500 [13]. NYUD [28], Multicue [29] and Pascal

[1]The code of DeepLab v2 is publicly available at `https://bitbucket.org/aquariusjay/deeplab-public-ver2`

VOC'12 [30] dataset. In the evaluation, we use two standard measures: F-measures ($\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$) when choosing an optimal dataset scale (ODS) which uses a fixed edge threshold for all images and an optimal image scale (OIS) which uses an adaptive threshold for each image [13]. We test the speed of different methods as well during experiments.

4.1. Datasets

We compare our method with competitive methods on four widely used datasets, including BSDS500 [13]. NYUD [28], Multicue [29] and Pascal VOC'12 [30] dataset. On the first three benchmarks, we evaluate our method for edge detection; on Pascal VOC'12 to evaluate it for object contour detection.

BSDS500 dataset has 500 images which are officially separated into 200 training, 100 validation and 200 test images. Each image has been manually annotated ground truth contours by 4-9 workers.

NYUD dataset contains 1449 densely labeled pairs of aligned RGB and depth images for indoor scene understanding, comprising 464 different indoor scenes across 26 scene classes. This dataset is split into 381 training, 414 validation and 654 testing images which have a 640×480 resolution. The dataset is used for evaluating boundary detection in [24, 28, 38, 39].

Multicue dataset proposed in motivated from psychophysics study is composed of short binocular video sequences of 100 challenging natural scenes captured by a stereo camera, in which the last image of each video is annotated for edge and boundary detection.

Pascal VOC'12 dataset for object contour detection [26]. Object contour detection is a higher-level computer vision task than edge detection. Unlike edge detection that responds to both foreground objects and background boundaries, it is intended to detect the contours of objects that belong to several interesting classes, such as 20 classes on Pascal VOC'12 dataset. Due to noisy annotations and requiring more semantic information, object contour detection on Pascal VOC'12 dataset is more challenging than edge detection on BSDS500 dataset.

4.2. BSDS500 dataset

On BSDS500 dataset, we utilize the training and validation sets for fine-tuning and evaluate methods on the standard test set. Data augmentation is same as [24]. Specifically, we rotate the images to 16 different angles and crop the largest rectangles in the rotated images. Meanwhile, we randomly flip images with a probability of 0.5 during training. We set the weight of the auxiliary loss to 0.5 and abandon this auxiliary branch to only use the well optimized master branch for final prediction on BSDS500 dataset.

4.2.1. Control experiments

In order to explore each component of our approach, we design two groups of control experiments on BSDS500 dataset. The first group is designed to evaluate the effectiveness of hybrid convolutional features and the auxiliary loss, described in Section 3.1 and 3.2. And we set HED [24] as the baseline for this group experiments.

Figure 2: The illustration of each component in our edge detection system. All the original images are from BSDS500 dataset [13]. From top to bottom, each row means original image, ground truth, HED (baseline), HCF (w/o auxiliary loss), HCF (with auxiliary loss, VGGNet), HCF (ResNet) and HCF (HybridNet) respectively. The demonstration shows that our method (HCF) is more confident on classifying edge/non-edge pixels than the baseline (HED). More details about these methods can be found in Section 4.2.1.

As shown in Table 1, hybrid convolutional features can more effectively fuse multi-level features than the baseline, and hybrid convolutional features lead a notable improvement by ODS F-measure of 1.4%. Adding an auxiliary loss is also useful for our method, but it only leads a small improvement by ODS F-measure of 0.2%. We find our method is more confident on classifying edge/non-edge pixels than the baseline, as shown in Figure 2. We consider the reason of this phenomenon is that different from the baseline which combines all side-outputs as the final edge map, our method directly predicts the final edge map using hybrid convolutional features.

The second group of control experiments is designed to evaluate different CNNs, including VGGNet (16 layers), ResNet (50 layers) and HybridNet. As shown in Table 1, ResNet version of HCF outperforms VGGNet version by only a small margin (ODS F-measure of 0.5%), although ResNet is much better than VGGNet on the ILSVRC CLS-LOC dataset. The reason of this phenomenon may be that ResNet is difficult to extract high-quality low-level features. The proposed HybridNet combines low-level features of VGGNet and high-level features of ResNet. When HybridNet is used to extract features, the performance of our method is much better (ODS F-measure of 1.0%)

Table 1: The first three rows are results of the first group of control experiments on BSDS500 dataset for evaluating the effectiveness of hybrid convolutional features and the auxiliary loss. The last three rows are results of the second group of control experiments on BSDS500 dataset for evaluating three different CNNs, including VGGNet, ResNet and HybridNet.

Method	ODS	OIS	FPS
HED (baseline)	.788	.808	**30**†
HCF (w/o auxiliary loss)	.802	.819	22†
HCF (with auxiliary loss)	**.804**	**.820**	22†
HCF (VGGNet)	.804	.820	**22**†
HCF (ResNet)	.809	.822	13†
HCF (HybridNet)	**.814**	**.827**	11†

than the one using VGGNet. The results support our supposition mentioned above and show the potentiality to train a fundamental CNN on the ILSVRC CLS-LOC dataset which is a good trade-off between extracting high-quality high- and low-level features for edge detection.

We also attempt to average the master branch output and the auxiliary output as the final edge map. However, we do not find any improvement on BSDS500 dataset.

4.2.2. Compared with competitive methods

In this section, we compare our method with 8 non-CNN-based methods, including Canny [12], EGB [40], MShift [41], gPb-UCM [13], Sketch Tokens [15], MCG [9], SE [16] and OEF [42], and 9 recent CNN-based methods, including N^4-Fields [17], DeepContour [22], DeepEdge [23], HFL [43], HED [24], RDS [25], CEDN [26], G-DSN [44] and COB [27].

As shown in Figure 3 and Table 2, our method achieves the state-of-the-art performance both in effectiveness and in efficiency. Two versions of our method, HCF and HybridNet version of HCF (HCF+HN), achieve ODS F-score of .804 and .814 respectively and are both better than human performance, which is known as ODS F-measure of .800. On BSDS500 dataset, where the size of testing images is 321×481, HCF achieves 22 frames per second (*fps*) and HCF+HN achieves 11 *fps* as well.

We observe that early CNN-based edge detectors, including N^4-Fields, DeepContour and DeepEdge, mostly predict edges in a patch-to-class paradigm, and the recent CNN-based methods, such as HFL, HED, CEDN and so on, infer edge maps in an image-to-image way. The latter methods are more efficient and effective than the former. It is because that the methods performing in an image-to-image way have larger receptive fields, shared and/or multi-level CNN features and fully convolutional architectures [24].

Both two versions of our method have an improvement than previous works. Except G-DSN+MS+VOC+NCuts [44], ODS F-measure of HCF and HCF+HN are 1.1% and 2.1% higher than the best previous methods. G-DSN+MS+VOC+NCuts also achieves comparable accuracy as ours. However, HCF and HCF+HN are much more efficient (22/11 *fps v.s* 1 *fps*) than this method. Besides, G-DSN+MS+VOC+NCuts is a complex edge detection system while our proposed methods are an end-to-end network without any post-processing. Edge detection is usually

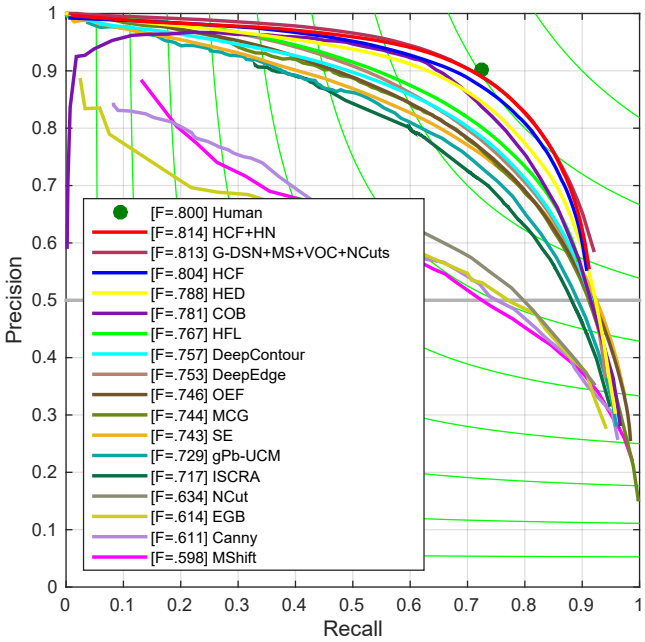

Figure 3: The evaluation results on BSDS500 dataset. HCF and HCF+HN achieve state-of-the-art F-score and are both higher than human performance (ODS F-score of .800). HN means utilizing HybridNet to extract features.

Table 2: The comparison with some competitors on BSDS500 [13] dataset (best viewed in color). The top three results are highlighted in red, green and blue respectively. HN means utilizing HybridNet to extract features; MS, VOC and NCuts mean ensemble testing, training with extra Pascal VOC Context dateset and post-processing with the Normalized Cuts; †means running on GPUs.

Method	ODS	OIS	FPS
Canny [12]	.611	.676	100
EGB [40]	.614	.658	10
MShift [41]	.598	.645	1/5
gPb-UCM [13]	.729	.755	1/240
Sketch Tokens [15]	.727	.746	1
MCG [9]	.744	.777	1/18
SE [16]	.743	.763	2.5
OEF [42]	.746	.770	2/3
N^4-Fields [17]	.753	.769	1/6†
DeepContour [22]	.757	.776	1/30†
DeepEdge [23]	.753	.772	1/1000†
HFL [43]	.767	.788	5/6†
HED [24]	.788	.808	30†
RDS [25]	.792	.810	30†
CEDN [26]	.788	.804	10†
G-DSN [44]	.789	.811	-
G-DSN+MS+VOC+NCuts [44]	.813	.831	1†
COB [27]	.793	.820	1†
HCF	.804	.820	22†
HCF+HN	.814	.827	11†

used as a basis technique in other vision tasks, so it should be simple and efficient. The proposed methods just fit this purpose and the efficiency make them easier to be involved in other vision systems.

4.3. NYUD dataset

on NYUD dataset, we utilize the training and validation sets for fine-tuning and test methods on the standard testing set. The images are rotated to 4 different angles (0, 90, 180 and 270 degrees) and horizontally flipped at each angle, so the augmented training set totally contains 6360 images.

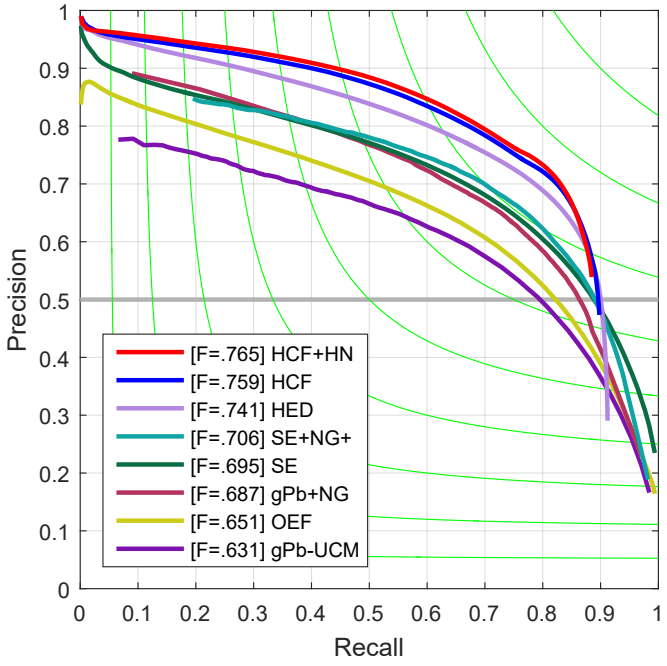

Figure 4: The evaluation results on NYUD dataset. HN means utilizing HybridNet to extract features.

Table 3: The comparison with some competitors on NYUD [28] dataset. **-RGB** means only using color information; **-HHA** means only using depth information; **-RGB-HHA** means using both color and depth information. HN means utilizing HybridNet to extract features; † means running on GPUs.

Method	ODS	OIS	FPS
gPb-UCM [13]	.632	.661	1/360
Silberman [28]	.658	.661	<1/360
gPb+NG [38]	.687	.716	1/375
SE [16]	.685	.699	5
SE+NG+ [39]	.710	.723	1/15
HED-HHA [24]	.681	.695	**20**†
HED-RGB [24]	.717	.732	**20**†
HED-RGB-HHA [24]	.741	.757	10†
HCF-HHA	.692	.705	13†
HCF-RGB	.738	.752	13†
HCF-RGB-HHA	.759	.773	7†
HCF+HN-HHA	.701	.715	7†
HCF+HN-RGB	.746	.760	7†
HCF+HN-RGB-HHA	**.765**	**.778**	4†

Following the success in SE+NG+ [39], FCN [20] and HED [24], we encode depth information as HHA feateres [39] in which the depth information is embedded into three channels, including horizontal disparity, height above ground and angle of the local surface normal with the inferred direction of gravity, which can be concatenated to construct HHA feature images. Then we separately train two edge detectors on color images and HHA feature images, and average the outputs of both models as the final results to make full use of both color information and depth information.

Table 4: Edge and boundary detection results on Multicue [29] dataset.

Method	ODS	OIS
Human-Boundary	.760 (0.017)	-
Multicue-Boundary [29]	.720 (0.014)	-
HED-Boundary [24]	.814 (0.011)	.822 (0.008)
HCF-Boundary	**.849 (0.016)**	**.859 (0.016)**
Human-Edge	.750 (0.024)	-
Multicue-Edge [29]	.830 (0.002)	-
HED-Edge [24]	.851 (0.014)	.864 (0.011)
HCF-Edge	**.858 (0.013)**	**.864 (.014)**

In the experiments on NYUD dataset, we set the weight of the auxiliary loss to 1 and average the master branch output and the auxiliary branch output as the final edge map. As used in HED [24], during evaluation we increase the maximum tolerance allowed for correct matches of edge predictions to ground truth from 0.0075 to 0.011.

Table 3 and Figure 4 show our method achieves the state-of-the-art performance on NYUD dataset. HCF is better than the best previous methods by 1.1%, 2.1% and 1.8% of ODS F-measure on HHA, RGB and RGB-HHA dataset respectively, and HCF+HN is also better than the best previous method by 2.0%, 2.9% and 2.4% of ODS F-measure.

4.4. Multicue dataset

Following [29, 24], we randomly split the dataset into 80 training and 20 testing images, and report the averaged F-measure over three times. We augment the training set by randomly horizontal flipping, rotating both images and their corresponding label maps to 4 angles (0, 90, 180, 270 degrees) and scaling images to 3 scales (75%, 100%, 125%). So the augmented set contains 1920 images for training. During training stage, we random crop 513 × 513 patches from original images.

In the experiments on Multicue dataset, we train HCF for only 2k iterations as in [24]. We set the weight of auxiliary loss to 1 and average the master branch output and the auxiliary branch output as the final edge map. During evaluation, the maximum tolerance allowed for correct matches of edge predictions to ground truth from 0.0075 increases to 0.011 as on NYUD dataset. We show evaluation results in Table 4. Our proposed method achieves a state-of-the-art performance for both boundary and edge detection as well and gives a remarkable improvement than HED.

4.5. Pascal VOC 2012 dataset

In this section, we evaluate the proposed method on Pascal VOC'12 [30] dataset for object contour detection [26]. Object contour detection is a higher-level computer vision task than

Table 5: The evaluation results on Pascal VOC'12 [30] dataset. HN means using HybridNet to extract features.

Method	ODS	OIS	FPS
MCG-BSDS500 [9]	.410	.486	1/18
LEP-BSDS500 [45]	.426	.517	1
HED [24]	.618	.650	**27**†
CEDN [26]	.653	.683	9†
HCF	.659	.689	17†
HCF+HN	**.677**	**.711**	10†

edge detection. Unlike edge detection that responds to both foreground objects and background boundaries, it is intended to detect the contours of the objects that belong to several interesting classes, such as 20 classes on Pascal VOC'12 dataset. Due to noisy annotations and requiring more semantic information, object contour detection on Pascal VOC'12 dataset is more challenging than edge detection on BSDS500 dataset.

We use the instance level annotation of Pascal VOC'12 dataset to extract object contours. The original PASCAL VOC'12 annotations leave a thin uncertain area between objects. We employ dense CRF to fill the uncertain area and refine ground truth contours as in [26]. During training we randomly flip the images and their corresponding label maps with a probability of 0.5.

In the experiments on Pascal VOC'12 dataset, we set the weight of the auxiliary loss to 1 and average the master branch output and the auxiliary branch output as the final edge map. During evaluation, we use the thinned annotated contours to evaluate all methods on the validation set, and increase the maximum tolerance allowed for correct matches of edge predictions to ground truth from 0.0075 to 0.01.

As shown in Table 5, our method achieves the state-of-the-art performance. HCF and HCF+HN are both better than the best previous method by ODS F-measure 0.6% and 2.4% respectively. The results demonstrate our method is suitable not only for low-level edge detection but for higher-level object contour detection. HCF+HN is better than HCF by 1.8% on Pascal VOC'12 dataset, but HCF+HN is better than HCF by only 1.0% on BSDS500 dataset. It demonstrates that powerful high-level information is more important to object contour detection than edge detection.

5. Conclusion

We have proposed a novel CNN-based edge detector, which can effectively fuse multi-level information in a feature-map based fashion and produce high-quality hybrid convolutional features (HCF) for detecting edges in an image-to-image way without any post-processing. To enhance the performance, we take the advantages of previous fundamental CNN architectures to simultaneously extract high- and low- level features from HybridNet which is developed from the VGGNet and ResNet. We have evaluated the proposed method on three widely used dataset for edge detection, including BSDS500, NYUD and MUlticue, and tested the proposed method on Pascal VOC'12 dataset for object contour detection. The results show that the proposed method obtains an improvement in performance over the state-of-the-art on all four datasets. In Figure 5, we demonstrate more results of the proposed method.

Although the efficient version of this method achieves 22*fps*, it is not a real-time edge detector. This will limit some real-time required applications, such as video edge detection and so on. In the future, we consider extending the proposed method to achieve a faster speed. To explore and design new fundamental CNNs on the ILSVRC CLS-LOC dataset which can achieve a good trade-off between extracting high-quality high- and low-level features is an attracted direction as well.

Acknowledgments

This research was supported by NSFC (NO. 61620106008, 61572264), CAST YESS Program, and IBM Global SUR award.

References

[1] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE CVPR, 2014, pp. 580–587. doi:10.1109/CVPR.2014.81.

[2] R. Girshick, Fast R-CNN, in: IEEE ICCV, 2015, pp. 1440–1448. doi:10.1109/ICCV.2015.169.

[3] N. Zeng, Z. Wang, H. Zhang, W. Liu, F. E. Alsaadi, Deep belief networks for quantitative analysis of a gold immunochromatographic strip, Cognitive Computation 8 (4) (2016) 684–692. doi:10.1007/s12559-016-9404-x.

[4] N. Zeng, H. Zhang, Y. Li, J. Liang, A. M. Dobaie, Denoising and deblurring gold immunochromatographic strip images via gradient projection algorithms, Neurocomputing 247 (Supplement C) (2017) 165 – 172. doi:10.1016/j.neucom.2017.03.056.

[5] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, A. M. Dobaie, Facial expression recognition via learning deep sparse autoencoders, Neurocomputing 273 (2018) 643–649. doi:10.1016/j.neucom.2017.08.043.

[6] H. Wang, J. Cao, X. Liu, J. Wang, T. Fan, J. Hu, Least-squares images for edge-preserving smoothing, Computational Visual Media 1 (1) (2015) 27–35. doi:10.1007/s41095-015-0004-6.

[7] P. Shao, S. Ding, L. Ma, Y. Wu, Y. Wu, Edge-preserving image decomposition via joint weighted least squares, Computational Visual Media 1 (1) (2015) 37–47. doi:10.1007/s41095-015-0006-4.

[8] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, Z. Tu, Hfs: Hierarchical feature selection for efficient image segmentation, in: ECCV, Springer, 2016, pp. 867–882. doi:10.1007/978-3-319-46487-9_53.

[9] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, J. Malik, Multiscale combinatorial grouping, in: IEEE CVPR, 2014, pp. 328–335. doi:10.1109/CVPR.2014.49.

[10] X. Liu, C. Li, T.-T. Wong, Boundary-aware texture region segmentation from manga, Computational Visual Media 3 (1) (2017) 61–71. doi:10.1007/s41095-016-0069-x.

[11] I. Sobel, Camera models and machine perception, Tech. rep., DTIC Document (1970).

[12] J. Canny, A computational approach to edge detection, IEEE TPAMI (6) (1986) 679–698. doi:10.1109/TPAMI.1986.4767851.

[13] P. Arbeláez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE TPAMI 33 (5) (2011) 898–916. doi:10.1109/TPAMI.2010.161.

[14] S. Konishi, A. L. Yuille, J. M. Coughlan, S. C. Zhu, Statistical edge detection: Learning and evaluating edge cues, IEEE TPAMI 25 (1) (2003) 57–74. doi:10.1109/TPAMI.2003.1159946.

[15] J. J. Lim, C. L. Zitnick, P. Dollár, Sketch tokens: A learned mid-level representation for contour and object detection, in: IEEE CVPR, 2013, pp. 3158–3165. doi:10.1109/CVPR.2013.406.

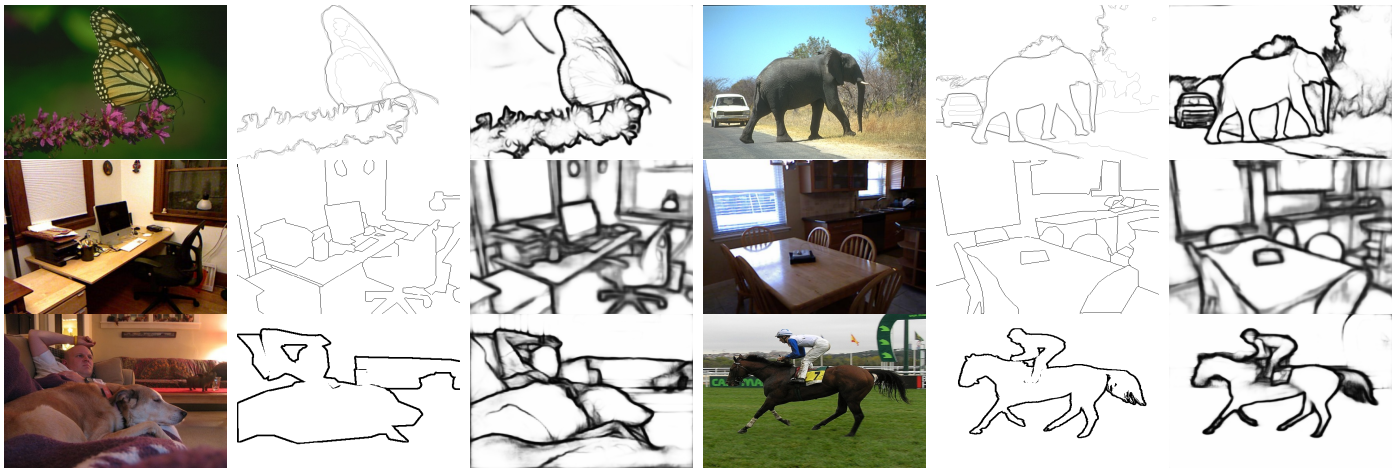

Figure 5: Results of the proposed method on three datasets. From top to bottom, each row means BSDS500, NYUD and Pascal VOC'12; HN means using HybridNet to extract features. HCF and HCF+HN are two versions of the proposed method, so we demonstrate three images of each version respectively.

[16] P. Dollár, C. L. Zitnick, Fast edge detection using structured forests, IEEE TPAMI 37 (8) (2015) 1558–1570. doi:10.1109/TPAMI.2014.2377715.

[17] Y. Ganin, V. Lempitsky, N^4-Fields: Neural network nearest neighbor fields for image transforms, in: ACCV, Springer, 2014, pp. 536–551. doi:10.1007/978-3-319-16808-1_36.

[18] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556. arXiv:1409.1556.

[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE CVPR, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[20] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: IEEE CVPR, 2015, pp. 3431–3440. doi:10.1109/CVPR.2015.7298965.

[21] 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 TPAMI PP (99) (2017) 1–1. doi:10.1109/TPAMI.2017.2699184.

[22] W. Shen, X. Wang, Y. Wang, X. Bai, Z. Zhang, DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection, in: IEEE CVPR, 2015, pp. 3982–3991. doi:10.1109/CVPR.2015.7299024.

[23] G. Bertasius, J. Shi, L. Torresani, DeepEdge: A multi-scale bifurcated deep network for top-down contour detection, in: IEEE CVPR, 2015, pp. 4380–4389. doi:10.1109/CVPR.2015.7299067.

[24] S. Xie, Z. Tu, Holistically-nested edge detection, in: IEEE ICCV, 2015, pp. 1395–1403. doi:10.1109/ICCV.2015.164.

[25] Y. Liu, M. S. Lew, Learning relaxed deep supervision for better edge detection, in: IEEE CVPR, 2016, pp. 231–240. doi:10.1109/CVPR.2016.32.

[26] J. Yang, B. Price, S. Cohen, H. Lee, M.-H. Yang, Object contour detection with a fully convolutional encoder-decoder network, in: IEEE CVPR, 2016. doi:10.1109/CVPR.2016.28.

[27] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, L. Van Gool, Convolutional oriented boundaries, in: ECCV, Springer, 2016, pp. 580–596.

[28] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: ECCV, Springer, 2012, pp. 746–760. doi:10.1007/978-3-642-33715-4_54.

[29] D. A. Mély, J. Kim, M. McGill, Y. Guo, T. Serre, A systematic comparison between visual cues for boundary detection, Vision research 120 (2016) 93–107.

[30] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[31] D. R. Martin, C. C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEE TPAMI 26 (5) (2004) 530–549. doi:10.1109/TPAMI.2004.1273918.

[32] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: NIPS, 2012, pp. 1097–1105. doi:10.1145/3065386.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE CVPR, 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[34] W. Liu, A. Rabinovich, A. C. Berg, Parsenet: Looking wider to see better, CoRR abs/1506.04579. arXiv:1506.04579.

[35] G. Huang, Z. Liu, K. Q. Weinberger, Densely connected convolutional networks, in: IEEE CVPR, 2017. doi:10.1109/CVPR.2017.243.

[36] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: ACM MM, 2014, pp. 675–678. doi:10.1145/2647868.2654889.

[37] Y. Bengio, X. Glorot, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of AISTATS, Vol. 9, 2010, pp. 249–256.

[38] S. Gupta, P. Arbelaez, J. Malik, Perceptual organization and recognition of indoor scenes from rgb-d images, in: IEEE CVPR, 2013, pp. 564–571. doi:10.1109/CVPR.2013.79.

[39] S. Gupta, R. Girshick, P. Arbeláez, J. Malik, Learning rich features from rgb-d images for object detection and segmentation, in: ECCV, Springer, 2014, pp. 345–360. doi:10.1007/978-3-319-10584-0_23.

[40] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation, IJCV 59 (2) (2004) 167–181. doi:10.1023/B:VISI.0000022288.19776.77.

[41] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE TPAMI 24 (5) (2002) 603–619. doi:10.1109/34.1000236.

[42] S. Hallman, C. C. Fowlkes, Oriented edge forests for boundary detection, in: IEEE CVPR, 2015, pp. 1732–1740. doi:10.1109/CVPR.2015.7298782.

[43] G. Bertasius, J. Shi, 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, 2015, pp. 504–512. doi:10.1109/ICCV.2015.65.

[44] I. Kokkinos, Pushing the boundaries of boundary detection using deep learning, CoRR abs/1511.07386. arXiv:1511.07386.

[45] Q. Zhao, Segmenting natural images with the least effort as humans., in: BMVC, 2015, pp. 110–1. doi:10.5244/C.29.110.