Noname manuscript No.
(will be inserted by the editor)

Holistically-Nested Edge Detection

Saining Xie · Zhuowen Tu

Received: date / Accepted: date

Abstract We develop a new edge detection algorithm that addresses two important issues in this long-standing vision problem: (1) holistic image training and prediction; and (2) multi-scale and multi-level feature learning. Our proposed method, holistically-nested edge detection (HED), performs image-to-image prediction by means of a deep learning model that leverages fully convolutional neural networks and deeply-supervised nets. HED automatically learns rich hierarchical representations (guided by deep supervision on side responses) that are important in order to resolve the challenging ambiguity in edge and object boundary detection. We significantly advance the state-of-the-art on the BSDS500 dataset (ODS F-score of 0.790) and the NYU Depth dataset (ODS F-score of 0.746), and do so with an improved speed (0.4s per image) that is orders of magnitude faster than some CNN-based edge detection algorithms developed before HED. We also observe encouraging results on other boundary detection benchmark datasets such as Multicue and PASCAL-Context.

1 Introduction

In this paper, we address the problem of detecting edges and object boundaries in natural images. This problem is both fundamental and of great importance to a variety of computer vision areas ranging from traditional tasks such as visual saliency, segmentation, object detection/recognition, tracking and motion analysis, medical imaging, structure-from-motion and 3D reconstruction, to modern applications like autonomous driving, mobile computing, and image-to-text analysis. It has been long understood that precisely localizing

9500 Gilman Drive
La Jolla, CA 92093-0515 USA
Tel.: +1-858-822-0908
Fax: +1-858-534-1128
E-mail: {s9xie,ztu}@ucsd.edu

edges in natural images involves visual perception of various "levels" (Hubel and Wiesel, 1962; Marr and Hildreth, 1980). A relatively comprehensive data collection and cognitive study (Martin et al, 2004) shows that while different human subjects do have somewhat different preferences regarding where to place the edges and boundaries, there was nonetheless impressive consistency between subjects, e.g. reaching an F-score 0.80 in the consistency study (Martin et al, 2004).

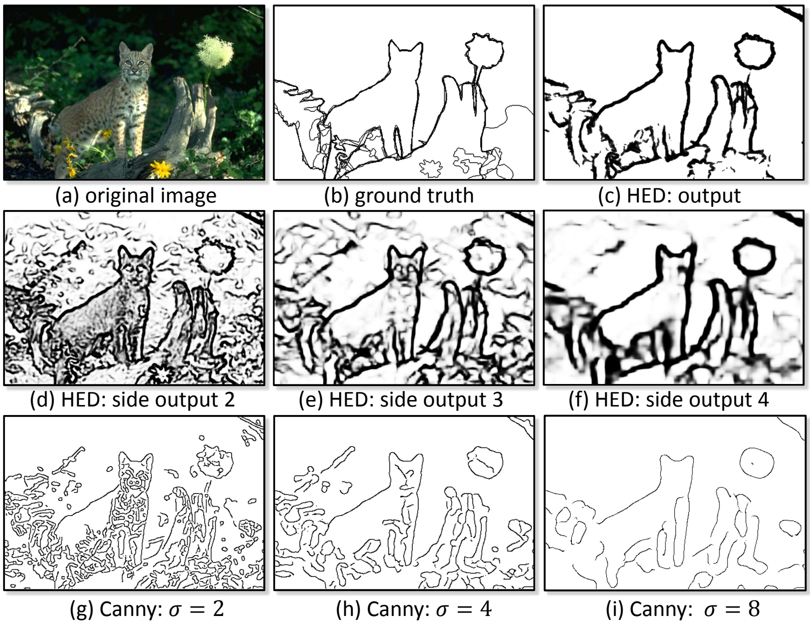

Fig. 1 Illustration of the proposed HED algorithm. In the first row: (a) shows an example test image in the BSDS500 dataset (Arbelaez et al, 2011); (b) shows its corresponding edges as annotated by human subjects; (c) displays the HED results. In the second row: (d), (e), and (f), respectively, show side edge responses from layers 2, 3, and 4 of our convolutional neural networks. In the third row: (g), (h), and (i), respectively, show edge responses from the Canny detector (Canny, 1986) at the scales $\sigma = 2.0$, $\sigma = 4.0$, and $\sigma = 8.0$. HED shows a clear advantage in consistency over Canny.

The history of computational edge detection is extremely rich; we now highlight a few representative works that have proven to be of great practical importance. Broadly speaking, one may categorize works into a few groups such as I: *early pioneering methods* like the Sobel detector (Kittler, 1983), zero-crossing (Marr and Hildreth, 1980; Torre and Poggio, 1986), and the widely adopted Canny detector (Canny, 1986); methods driven by II: *information theory* on top of features arrived at through careful manual design, such as Statistical Edges (Konishi et al, 2003), Pb (Martin et al, 2004), and gPb (Arbelaez et al, 2011); and III: *learning-based* methods that remain reliant on features of human design, such as BEL (Dollár et al, 2006), Multi-scale (Ren, 2008), Sketch Tokens (Lim et al, 2013), and Structured Edges (Dollár and Zitnick, 2015). In addition, there has been a recent wave of development using *Convolutional Neural Networks* that emphasize the importance of automatic hierarchical feature learning, including N^4-Fields (Ganin and Lempitsky, 2014), DeepContour (Shen et al, 2015), DeepEdge (Bertasius et al, 2015), and CSCNN (Hwang and Liu, 2015). Prior to this explosive development in deep learning, the Structured Edges method (typically abbreviated SE) (Dollár and Zitnick, 2015) emerged as one of the most celebrated systems for edge detection, thanks to its state-of-the-art performance on the BSDS500 dataset (Martin et al, 2004; Arbelaez et al, 2011) (with, e.g., F-score of 0.746) and its practically significant speed of 2.5 frames per second.

Convolutional neural networks (CNN) (LeCun et al, 1989) have achieved a great success in automatically learning thousands (or even millions or billions) of features for pattern recognition, under the paradigm of image-to-class classification (e.g. predicting which category an image belongs to (Russakovsky et al, 2014)) or patch-to-class classification (e.g predicting which object an image patch contains (Girshick et al, 2014)). CNN-based edge detection methods before HED (Ganin and Lempitsky, 2014; Shen et al, 2015; Bertasius et al, 2015; Hwang and Liu, 2015) mostly follow a patch-to-class paradigm, which is patch-centric. These patch-centric approaches fall into the category of "sliding-window" methods that perform prediction by considering dense, overlapping windows of the image, often centered at every pixel; this creates a big bottleneck in both training and testing; for example, time to detect edges in one static image for these methods ranges from several seconds (Ganin and Lempitsky, 2014) to a few hours (Bertasius et al, 2015) (even when using modern GPUs). Recently proposed fully convolutional neural networks (FCN) (Long et al, 2015), targeted for the task of semantic image labeling, instead points to a promising direction of performing training/testing for the entire image altogether, which is under a image-centric paradigm. Applying FCN to the edge detection problem however produces an unsatisfactory result (e.g. F-score 0.745 on BSDS500) as edges observe strong multi-scale aspects that is quite different from semantic labeling. In this regard, the deeply-supervised nets method (DSN) (Lee et al, 2015) provides a principled and clean solution for multi-scale learning and fusion where supervised information is jointly enforced in the individual convolutional layers during training.

Motivated by fully convolutional networks (Long et al, 2015) and deeply-supervised nets (Lee et al, 2015), we develop an end-to-end edge detection system, holistically-nested edge detection (HED), that automatically learns the type of rich hierarchical features that are crucial if we are to approach the human ability to resolve ambiguity in natural image edge and object boundary detection. We use the term "holistic", because HED, despite not explicitly modeling structured output, aims to train and predict edges in an image-to-image fashion. With "nested", we emphasize the inherited and progressively refined edge maps produced as side outputs: we intend to show that the path along which each prediction is made is common to each of these edge maps, with successive edge maps being more concise. This integrated learning of hierarchical features is in distinction to previous multi-scale approaches (Witkin, 1984; Yuille and Poggio, 1986; Ren, 2008) in which scale-space edge fields are neither automatically learned nor hierarchically connected. We find that the favorable characteristics of these underlying techniques manifest in HED being both accurate and computationally efficient. Figure 1 gives an illustration of an example image together with the human subject ground truth annotation, as well as results by the proposed HED edge detector (including the side responses of the individual layers), and results by the Canny edge detector (Canny, 1986) with different scale parameters. Not only are Canny edges at different scales not directly connected, they also exhibit spatial shift and inconsistency.

Methods after HED: After the acceptance of the conference version of our work (Xie and Tu, 2015), HED has been extended to new applications and applied in different domains: an edge detector is trained using supervised labeling information automatically obtained from videos using motion cues (Li et al, 2016); a weakly-supervised learning strategy is proposed in (Khoreva et al, 2016) to reduce the burden in obtaining a large amount of training labels; further improvement on the BSDS500 dataset is achieved in (Kokkinos, 2016) by carefully fusing multiple cues; boundary detection methods towards extracting high-level semantics have been proposed in (Zhu et al, 2015; Chen et al, 2015; Premachandran et al, 2015); extension and refinement to 3D Vascular boundaries in medical imaging is developed in (Merkow et al, 2016); scale-sensitive deep supervision is introduced in (Shen et al, 2016) for object skeleton extraction.

2 Significance and Related Work

The proposed holistically-nested edge detector (HED) tackles two critical issues: (1) holistic image training and prediction, inspired by fully convolutional neural networks (Long et al, 2015), for image-to-image classification (the system takes an image as input, and directly produces the edge map image as output); and (2) nested multi-scale feature learning, inspired by deeply-supervised nets (Lee et al, 2015), that performs deep layer supervision to “guide” early classification results. We discuss below the significance of the proposed HED algorithm when compared with the existing algorithms along two directions in terms of: (1) edge and object boundary detection; and (2) multi-scale learning in neural networks.

2.1 Edge and object boundary detection

The task of edge and object boundary detection is inherently challenging. After decades of research, there have emerged a number of properties that are key and that are likely to play a role in a successful system: (1) carefully designed and/or learned features (Martin et al, 2004; Dollár et al, 2006), (2) multi-scale response fusion (Witkin, 1984; Ruderman and Bialek, 1994; Ren, 2008), (3) engagement of different levels of visual perception (Hubel and Wiesel, 1962; Marr and Hildreth, 1980; Van Essen and Gallant, 1994; Hou et al, 2013) such as mid-level Gestalt law information (Elder and Goldberg, 2002), (4) incorporating structural information (intrinsic correlation carried within the input data and output solution) (Dollár and Zitnick, 2015) and context (both short- and long- range interactions) (Tu, 2008), (5) making holistic image predictions (referring to approaches that perform prediction by taking the image contents globally and directly) (Liu et al, 2011), (6) exploiting 3D geometry (Hoiem et al, 2008), and (7) addressing occlusion boundaries (Hoiem et al, 2007).

Structured Edges (SE) (Dollár and Zitnick, 2015) primarily focuses on three of these aspects: using a large number of manually designed features (property 1), fusing multi-scale responses (property 2), and incorporating structural information (property 4). A recent wave of work using CNN for patch-based edge prediction (Ganin and Lempitsky, 2014; Shen et al, 2015; Bertasius et al, 2015; Hwang and Liu, 2015) contains an alternative common thread that focuses on three aspects: automatic feature learning (property 1), multi-scale response fusion (property 2), and possible engagement of different levels of visual perception (property 3). However, due to the lack of deep supervision (that we include in our method), the multi-scale responses produced at the hidden layers in (Bertasius et al, 2015; Hwang and Liu, 2015) are less semantically meaningful, since feedback must be back-propagated through the intermediate layers. More importantly, their patch-to-pixel or patch-to-patch strategy results in significantly downgraded training and prediction efficiency.

By "holistically-nested", we intend to emphasize that we are producing an end-to-end edge detection system, a strategy inspired by fully convolutional neural networks (Long et al, 2015), but with additional deep supervision on top of trimmed VGG nets (Simonyan and Zisserman, 2015) (shown in Figure 3). In the absence of deep supervision and side outputs, a fully convolutional network (Long et al, 2015) (FCN) produces a less satisfactory result (e.g. F-score 0.745 on BSDS500) than HED, since edge detection demands highly accurate edge pixel localization. One thing worth mentioning is that our image-to-image training and prediction strategy still has not explicitly engaged contextual information, since constraints on the neighboring pixel labels are not directly enforced in HED. In addition to the speed gain over patch-based CNN edge detection methods, the performance gain is largely due to three aspects: (1) FCN-like image-to-image training allows us to simultaneously train on a significantly larger amount of samples (see Table 5); (2) deep supervision in our model guides the learning of more transparent features (see Table 2); (3) interpolating the side outputs in the end-to-end learning encourages coherent contributions from each layer (see Table 4).

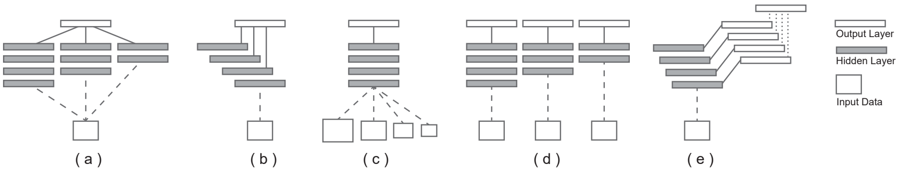

Fig. 2 Illustration of different multi-scale deep learning architecture configurations: (a) multi-stream architecture; (b) skip-layer net architecture; (c) a single model running on multi-scale inputs; (d) separate training of different networks; (e) our proposed holistically-nested architectures, where multiple side outputs are added.

2.2 Multi-scale learning in neural networks

In this section, we discuss related neural-network-based approaches, particularly those that emphasize multi-scale feature learning. Due to the nature of hierarchical learning in the deep convolutional neural networks, the concept of multi-scale and multi-level learning might differ from situation to situation. For example, multi-scale learning can be “inside” the neural network, in the form of increasingly larger receptive fields and downsampled (strided) layers. In this “inside” case, the feature representations learned in each layer are naturally multi-scale. On the other hand, multi-scale learning can be “outside” of the neural network, for example by “tweaking the scales” of input images. While these two variants have some notable similarities, we have seen both of them applied to various tasks.

We continue by next formalizing the possible configurations of multi-scale deep learning into four categories, namely, *multi-stream* learning, *skip-net* learning, a *single model* running on multiple inputs, and training of *independent* networks. An illustration is shown in Fig 2. Having these possibilities in mind will help make clearer the ways in which our proposed *holistically-nested* network approach differs from previous efforts and will help to highlight the important benefits in terms of representation and efficiency.

Multi-stream learning (Buyssens et al, 2013; Neverova et al, 2014) A typical multi-stream learning architecture is illustrated in Fig 2(a). Note that the multiple (parallel) network streams have different parameter numbers and receptive field sizes, corresponding to multiple scales. Input data are simultaneously fed into multiple streams, after which the concatenated feature responses produced by the various streams are fed into a global output layer to produce the final result.

Skip-layer network learning: Examples of this form of network include (Long et al, 2015; Hariharan et al, 2015; Bertasius et al, 2015; Sermanet et al, 2012; Ganin and Lempitsky, 2014). The key concept in “skip-layer” network learning is shown in Fig 2(b). Instead of training multiple parallel streams, the topology for the skip-net architecture centers on a primary stream. Links are added to incorporate the feature responses from different levels of the primary network stream, and these responses are then combined in a shared output layer.

A common point in the two settings above is that, in both of the architectures, there is only one output loss function with a single prediction produced. However, in edge detection, it is often favorable (and indeed prevalent) to obtain multiple predictions to combine the edge maps together.

Single model on multiple inputs: To get multi-scale predictions, one can also run a single network (or networks with tied weights) on multiple (scaled) input images, as illustrated in Fig 2(c). This strategy can happen at both the training stage (as data augmentation) and at the testing stage (as “ensemble testing”). One notable example is the tied-weight pyramid networks (Farabet et al, 2013). This approach is also common in non-deep-learning based methods (Dollár and Zitnick, 2015). Note that ensemble testing impairs the prediction efficiency of learning systems, especially with deeper models(Bertasius et al, 2015; Ganin and Lempitsky, 2014).

Training independent networks: As an extreme variant to Fig 2(a), one might pursue Fig 2(d), in which multi-scale predictions are made by training multiple independent networks with different depths and different output loss layers. This might be practically challenging to implement as this duplication would multiply the amount of resources required for training.

Holistically-nested networks: We list these variants to help clarify the distinction between existing approaches and our proposed holistically-nested network approach, illustrated in Fig 2(e). There is often significant redundancy in existing approaches, in terms of both representation and computational complexity. Our proposed holistically-nested network is a relatively simple variant that is able to produce predictions from multiple scales. The architecture can be interpreted as a “holistically-nested” version of the “independent networks” approach in Fig 2(d), motivating our choice of name. Our architecture comprises a single-stream deep network with multiple side outputs. This architecture resembles several previous works, particularly the deeply-supervised net(Lee et al, 2015) approach in which the authors show that hidden layer supervision can improve both optimization and generalization for image classification tasks. The multiple side outputs also give us the flexibility to add an additional fusion layer if a unified output is desired.

3 Our Approach and Formulation

In this section, we describe in detail our our proposed HED edge detection system and start by introducing the formulation first.

3.1 Formulation

In this section, We give the formulation of HED and discuss in detail the training and testing procedure, as well as the network structures of HED.

3.1.1 Training

We denote our input training data set by $S = \{(X_n, Y_n), n = 1, \dots, N\}$, where sample $X_n = \{x_j^{(n)}, j = 1, \dots, |X_n|\}$ denotes the raw input image and $Y_n = \{y_j^{(n)}, j = 1, \dots, |X_n|\}, y_j^{(n)} \in \{0, 1\}$ denotes the corresponding ground truth binary edge map for image X_n. We subsequently drop the subscript n for notational simplicity, since we consider each image holistically and independently. Our goal is to have a network that learns features from which it is possible to produce edge maps approaching the ground truth. For simplicity, we denote the collection of all standard network layer parameters as $\mathbf{W}$. Suppose in the network we have M side-output layers. Each side-output layer is also associated with a classifier, in which the corresponding weights are denoted as $\mathbf{w} = (\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(M)})$. We consider the objective function

$$\mathcal{L}_{\text{side}}(\mathbf{W}, \mathbf{w}) = \sum_{m=1}^{M} \alpha_m \ell_{\text{side}}^{(m)}(\mathbf{W}, \mathbf{w}^{(m)}), \tag{1}$$

where ℓ_{side} denotes the image-level loss function for side-outputs. In our image-to-image training, the loss function is computed over all pixels in a training image $X = (x_j, j = 1, \dots, |X|)$ and edge map $Y = (y_j, j = 1, \dots, |X|), y_j \in \{0, 1\}$. For a typical natural image, the distribution of edge/non-edge pixels is heavily biased: 90% of the ground truth is non-edge. A cost-sensitive loss function is proposed in (Hwang and Liu, 2015), with additional trade-off parameters introduced for biased sampling.

We instead use a simpler strategy to automatically balance the loss between positive/negative classes. We introduce a class-balancing weight β on a per-pixel term basis. Index j is over the image spatial dimensions of image X. Then we use this class-balancing weight as a simple way to offset this imbalance between edge and non-edge. Specifically, we define the following class-balanced cross-entropy loss function used in Equation (1)

$$\begin{aligned} \ell_{\text{side}}^{(m)}(\mathbf{W}, \mathbf{w}^{(m)}) = &-\beta \sum_{j \in Y_+} \log \Pr(y_j = 1 | X; \mathbf{W}, \mathbf{w}^{(m)}) \\ &-(1-\beta) \sum_{j \in Y_-} \log \Pr(y_j = 0 | X; \mathbf{W}, \mathbf{w}^{(m)}) \end{aligned} \tag{2}$$

where $\beta = |Y_-|/|Y|$ and $1 - \beta = |Y_+|/|Y|$. $|Y_-|$ and $|Y_+|$ denote the edge and non-edge ground truth label sets, respectively. $\Pr(y_j = 1|X; \mathbf{W}, \mathbf{w}^{(m)}) = \sigma(a_j^{(m)}) \in [0, 1]$ is computed using sigmoid function $\sigma(.)$ on the activation value at pixel j. At each side output layer, we then obtain edge map predictions $\hat{Y}_{\text{side}}^{(m)} = \sigma(\hat{A}_{\text{side}}^{(m)})$, where $\hat{A}_{\text{side}}^{(m)} \equiv \{a_j^{(m)},\ j = 1, \dots, |Y|\}$ are activations of the side-output of layer m.

To directly utilize side-output predictions, we add a "weighted-fusion" layer to the network and (simultaneously) learn the fusion weight during training. Our loss function at the fusion layer $\mathcal{L}_{\text{fuse}}$ becomes

$$\mathcal{L}_{\text{fuse}}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = \text{Dist}(Y,\ \hat{Y}_{\text{fuse}}) \tag{3}$$

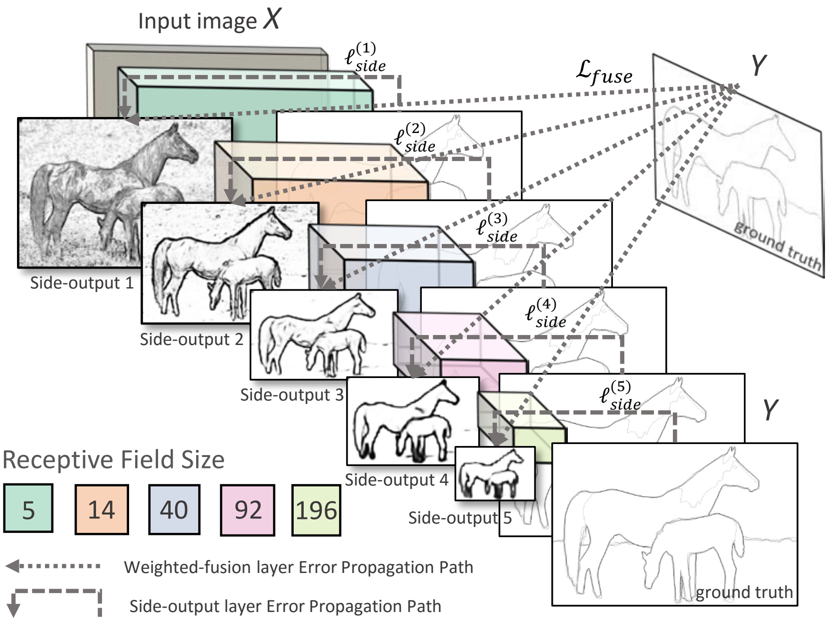

Fig. 3 Illustration of our network architecture for edge detection, highlighting the error backpropagation paths. Side-output layers are inserted after convolutional layers. Deep supervision is imposed at each side-output layer, guiding the side-outputs towards edge predictions with the characteristics we desire. The outputs of HED are multi-scale and multi-level, with the side-output-plane size becoming smaller and the receptive field size becoming larger. One weighted-fusion layer is added to automatically learn how to combine outputs from multiple scales. The entire network is trained with multiple error propagation paths (dashed lines).

where $\hat{Y}_{\text{fuse}} \equiv \sigma(\sum_{m=1}^{M} h_m \hat{A}_{\text{side}}^{(m)})$ where $\mathbf{h} = (h_1, \ldots, h_M)$ is the fusion weight. $\text{Dist}(\cdot, \cdot)$ is the distance between the fused predictions and the ground truth label map, which we set to be cross-entropy loss. Putting everything together, we minimize the following objective function via standard (back-propagation) stochastic gradient descent:

$$(\mathbf{W}, \mathbf{w}, \mathbf{h})^{\star} = \text{argmin}(\mathcal{L}_{\text{side}}(\mathbf{W}, \mathbf{w}) + \mathcal{L}_{\text{fuse}}(\mathbf{W}, \mathbf{w}, \mathbf{h})) \tag{4}$$

See section 4 for detailed hyper-parameter and experiment settings.

3.1.2 Testing

During testing, given image X, we obtain edge map predictions from both the side output layers and the weighted-fusion layer:

$$(\hat{Y}_{\text{fuse}},\ \hat{Y}_{\text{side}}^{(1)},\ \ldots,\ \hat{Y}_{\text{side}}^{(M)}) = \text{CNN}(X, (\mathbf{W}, \mathbf{w}, \mathbf{h})^{\star}), \tag{5}$$

where CNN(·) denotes the edge maps produced by our network. The final unified output can be obtained by further aggregating these generated edge maps. The details will be discussed in section 4.

$$\hat{Y}_{\text{HED}} = \text{Average}(\hat{Y}_{\text{fuse}},\ \hat{Y}^{(1)}_{\text{side}},\ \ldots,\ \hat{Y}^{(M)}_{\text{side}}) \tag{6}$$

3.2 Network Architecture

Next, we describe the network architecture of HED.

3.2.1 Trimmed network for edge detection

The choice of hierarchy for our framework deserves some thought. We need the architecture (1) to be deep, so as to efficiently generate perceptually multi-level features; and (2) to have multiple stages with different strides, so as to capture the inherent scales of edge maps. We must also keep in mind the potential difficulty in training such deep neural networks with multiple stages when starting from scratch. Recently, VGGNet (Simonyan and Zisserman, 2015) has been seen to achieve state-of-the-art performance in the ImageNet challenge, with great depth (16 convolutional layers), great density (stride-1 convolutional kernels), and multiple stages (five 2-stride downsampling layers). Recent work (Bertasius et al, 2015) also demonstrates that fine-tuning deep neural networks pre-trained on the general image classification task is useful to the low-level edge detection task. We therefore adopt the VGGNet architecture but make the following modifications: (a) we connect our side output layer to the last convolutional layer in each stage, respectively conv1_2, conv2_2, conv3_3, conv4_3, conv5_3. The receptive field size of each of these convolutional layers is identical to the corresponding side-output layer; (b) we cut the last stage of VGGNet, including the 5th pooling layer and all the fully connected layers. The reason for "trimming" the VGGNet is two-fold. First, because we are expecting meaningful side outputs with different scales, a layer with stride 32 yields a too-small output plane with the consequence that the interpolated prediction map will be too fuzzy to utilize. Second, the fully connected layers (even when recast as convolutions) are computationally intensive, so that trimming layers from pool5 on can significantly reduce the memory/time cost during both training and testing. Our final HED network architecture has 5 stages, with strides 1, 2, 4, 8 and 16, respectively, and with different receptive field sizes, all nested in the VGGNet. See Table 1 for a summary of the configurations of the receptive fields and strides.

3.2.2 Architecture alternatives

Below we discuss some possible alternatives in architecture design, and in particular, the role of deep supervision of HED for the edge detection task.
FCN and skip-layer architecture

Table 1 The receptive field and stride size in VGGNet (Simonyan and Zisserman, 2015) used in HED. The bolded convolutional layers are linked to additional side-output layers.

layer	**c1_2**	p1	**c2_2**	p2	**c3_3**
rf size	**5**	6	**14**	16	**40**
stride	**1**	2	**2**	4	**4**
layer	p3	**c4_3**	p4	**c5_3**	(p5)
rf size	44	**92**	100	**196**	212
stride	8	**8**	16	**16**	32

Table 2 Performance of alternative architectures on BSDS500 dataset. The "fusion-output without deep supervision" result is learned w.r.t Eqn. 3. The "fusion-output with deep supervision" result is learned w.r.t. to Eqn. 4.

	ODS	OIS	AP
FCN-8S	.697	.715	.673
FCN-2S	.738	.756	.717
Fusion-output (w/o deep supervision)	.771	.785	.738
Fusion-output (with deep supervision)	.782	.802	.787

The topology used in the FCN model differs from that in our HED model in several aspects. As we have discussed, while FCN reinterprets classification nets for per-pixel prediction, it has only one output loss function. Thus, in FCN, although the skip net structure is a DAG that combines coarse, high-layer information with fine low-layer information, it does not explicitly produce multi-scale output predictions. We explore how this architecture can be used for the edge detection task under the same experimental setting as our HED model. We first try to directly apply the FCN-8s model by replacing the loss function with cross-entropy loss for edge detection. The results shown in first row of Table 2 are unsatisfactory, which is expected since this architecture is still not fine enough. We further explore whether the performance can be improved by adding even more links from low-level layers. We then create an FCN-2s network that adds additional links from the pool1 and pool2 layers. Still, directly applying the FCN skip-net topology falls behind our proposed HED architecture (see second row of Table 2). With heavy tweaking of FCN, there is a possibility that one might be able to achieve competitive performance on edge detection, but the multi-scale side-outputs in HED are seen to be natural and intuitive for edge detection.

The role of deep supervision

Since we incorporate a weighted-fusion output layer that connects each side-output layer, there is a need to justify the adoption of the deep supervision terms (specifically, $\ell_{\text{side}}(\mathbf{W}, \mathbf{w}^{(m)})$: now the entire network is path-connected and the output-layer parameters can be updated by back-propagation through the weighted-fusion layer error propagation path (subject to Equation 3). Here we show that deep supervision is important to obtain desired edge maps. The key characteristic of our proposed network is that each network layer is supposed to play a role as a singleton network responsible for producing an

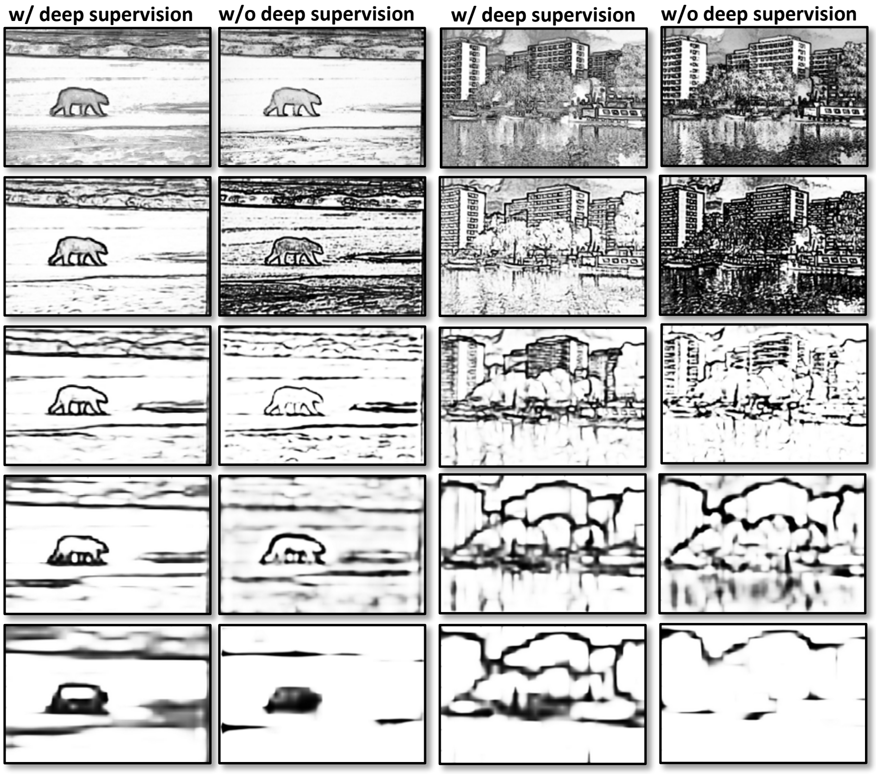

Fig. 4 Two examples illustrating how deep supervision helps side-output layers to produce multi-scale dense predictions. Note that in the left column, the side outputs become progressively coarser and more "global", while critical object boundaries are preserved. In the right column, the predictions tends to lack any discernible order (e.g. in layers 1 and 2), and many boundaries are lost in later stages.

edge map at a certain scale. Here are some qualitative results based on the two variants discussed above: (1) training with both weighted-fusion supervision and deep supervision, and (2) training with weighted-fusion supervision only. We observe that with deep supervision, the nested side-outputs are natural and intuitive, insofar as the successive edge map predictions are progressively coarse-to-fine, local-to-global. On the other hand, training with only the weighted-fusion output loss gives edge predictions that lack such discernible order: many critical edges are absent at the higher layer side output; under exactly same experimental setup, the result on the benchmark dataset (row three of Table 2) differs only marginally in F-score but displays severely degenerated average precision; without direct control and guidance across multiple scales, this network is heavily biased towards learning large structure edges.

4 Experiments

In this section we discuss our detailed implementation and report the performance of our proposed algorithm. We experiment HED on four benchmark datasets including BSDS500 (Arbelaez et al, 2011), NYUD (Silberman et al, 2012), Multicue-edge/boundary (Mély et al, 2015) and PASCAL-Context (Everingham et al, 2014). Some qualitative results are shown in Figure 7.

4.1 Implementation

We implement our framework using the publicly available *Caffe* Library and build on top of the publicly available implementations of FCN(Long et al, 2015) and DSN(Lee et al, 2015). Thus, relatively little engineering hacking is required. In our HED system, the network is fine-tuned from the VGG-16 Net model (Simonyan and Zisserman, 2015)

Model parameters. In contrast to fine-tuning CNN for image classification or semantic segmentation, adapting CNN for low-level edge detection requires special care. Even with initialization from a pre-trained model, sparse ground truth distributions coupled with conventional loss functions lead to difficulties in network convergence. Following the strategies outlined in (Dollár and Zitnick, 2015), we evaluated various network modification as well as training hyper-parameters on a validation set. Through experimentation, we choose the following hyper-parameters: mini-batch size (10), learning rate (1e-6), loss-weight α_m for each side-output layer (1), momentum (0.9), nested filter initialization weights (0), fusion layer initialization weights (1/5), weight decay (0.0002), training iterations (10,000; divide learning rate by 10 after 5,000). We found that these hyper-parameters led to the best performance and deviations in F-score on the validation set tended to be very small. We also investigated the use of additional nonlinearities by adding an additional layer (with 50 filters and a ReLU) prior to each side-output layer; and that this decreased performance. We also observed that nested multi-scale framework is insensitive to input image scales so, we train on full-resolution images without any resizing or cropping. In the experiments that follow, we fix the values of all hyper-parameters discussed above and concentrate on benefits of specific variants of HED.

Consensus sampling. In our approach, we duplicate the ground truth at each side-output layer and resize the (downsampled) side output to its original scale creating a mismatch in the high-level side-outputs. The edge predictions are coarse and global, while the ground truth still contains many weak edges that could be considered as noise. This leads to problematic convergence behavior, even with the help of a pre-trained model. We observe that this mismatch leads to gradients that explode at the high-level side-output layers. Therefore, we adjust ground truth labels in the BSDS500 dataset to combat this issue. Specifically, the ground truth labels are provided by multiple annotators so greater labeler consensus indicates stronger ground truth edges. During training, we use majority voting to obtain the positive labels. For example, on BSDS500 dataset, where each image was segmented by five different subjects on average, we assign a pixel to be positive if and only if it is labeled as positive by *at least three* annotators. All other labeled pixels are casted into negatives. This greatly helps convergence in the side-output layers. For low level layers, this consensus approach brings additional regularization to edge point classification and prevents the network from being distracted by weak edges. Although not fully explored in our paper, a careful handling of consensus levels of ground truth edges might lead to further improvement.

Data augmentation. Data augmentation has proven to be a crucial technique in deep networks. We rotate the images to 16 different angles and crop the largest rectangle in the rotated image; we also flip the image at each angle, leading to an augmented training set that is a factor of 32 larger than the unaugmented set. After our conference paper (Xie and Tu, 2015), we add additional augmentation by scaling the training images to 50%, 100%, 150% of its original size. Though holistically-nested networks naturally handle the multi-scale feature learning with its architecture design, we found that the scale augmentation improves the edge detection results from ODS=0.782 to ODS=0.790. During testing we operate on an input image at its original size. We also note that "ensemble testing" (making predictions on rotated/flipped images and averaging the predictions) yields no improvements in F-score, nor in average precision.

Different pooling functions. Previous work (Bertasius et al, 2015) suggests that different pooling functions can have a major impact on edge detection results. We conduct a controlled experiment in which all pooling layers are replaced by average pooling. We find that using average pooling decrease the performance to ODS=.741.

In-network bilinear interpolation. Side-output prediction upsampling is implemented with in-network deconvolutional layers, similar to those in (Long et al, 2015). We fix all the deconvolutional layers to perform linear interpolation. Although it was pointed out in (Long et al, 2015) that one can learn arbitrary interpolation functions, we find that learned deconvolutions provide no noticeable improvements in our experiments.

Running time. Training takes about 7 hours on a single NVIDIA K40 GPU. HED produces an edge response for an image of size 320 × 480 or 480 × 320 in about 400 ms (including the interface overhead). This is significantly more efficient than existing CNN-based methods (Shen et al, 2015; Bertasius et al, 2015).

Table 3 Comparisons with training independent networks. Performance measured on NVIDIA Titan X GPU.

	Time (ms)	Forward (ms)	Backward (ms)
HED-fusion	186.002	60.664	125.266
Independent Networks	501.105	134.064	366.944

Training independent networks. Our approach builds upon configuration D in Figure 2, however we achieve the same or better performance by using a nested multi-scale model that is also faster and more compact. As a proof-of-concept and sanity check, we train 5 independent networks induced from the five convolutional blocks of VGG-net. Those networks are of different depths, where the number of convolutional layers are 2, 4, 7, 10, 13, respectively. During test, we average the outputs of these individual networks as the final prediction. These independent networks achieve an ODS=0.784 where-as HED achieves an ODS=0.790 under the same experimental conditions. Comparing the speed of each approach by clocking neural network iteration time, illustrates another advantage of HED over independent networks. Unsurprisingly, HED is significantly faster. Table 3 summarizes the forward, backward, and overall iteration time, averaged over 50 iterations. As shown in the table, HED achieves the better ODS score with a 2.7x speed-up compared to the brute-force ensemble of independent networks.

4.2 BSDS500 dataset

We evaluate HED on the Berkeley Segmentation Dataset and Benchmark (BSDS500) (Arbelaez et al, 2011). BSDS500 is composed of 200 training, 100 validation, and 200 testing images where each image is manually annotated ground truth contours. Edge detection accuracy is evaluated using three standard measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP). We apply a standard non-maximal suppression technique to our edge maps to obtain thinned edges for evaluation.

First, we report the results of the original HED system (Xie and Tu, 2015) in which an input image is resized to a fixed size of 400 × 400. We name this algorithm HED (400 × 400). Second, we report improved results of training HED by preserving the image aspect ratio and performing additional data augmentation. These results are show in Table 5 and Figure 5.

Side outputs. To explicitly validate the side outputs, we summarize the results produced by the individual side-outputs at different scales in Table 4, including different combinations of the multi-scale edge maps. We emphasize here that all the side-output predictions are obtained in one pass; this enables us to fully investigate different configurations of combining the outputs at no extra cost. There are several interesting observations from the results: for instance, combining predictions from multiple scales yields better performance; moreover, all the side-output layers contribute to performance gain, either in F-score or averaged precision. We see this in Table 4, where the side-output layer 1 and layer 5 (the lowest and highest layers) achieve similar relatively low performance. One might conclude that these two side-output layers would not be useful in the averaged results,however, this turns out to false. The average of side-outputs 1-4 achieves ODS=0.760, but incorporating side-output (5) increases performance to ODS=0.774. We find similar phenomenon when considering other ranges. As mentioned above, the predictions obtained using

Table 4 Illustration of single and averaged side output in HED on the BSDS500 dataset as a running example. The individual side output contributes to the fused/averaged result. Note that the learned weighted-fusion (*Fusion-output*) achieves best F-score, while directly averaging all of the five layers (*Average 1-5*) produces better average precision. Merging those two readily available outputs further boost the performance.

	ODS	OIS	AP
Side-output 1	.595	.620	.582
Side-output 2	.697	.715	.673
Side-output 3	.738	.756	.717
Side-output 4	.740	.759	.672
Side-output 5	.606	.611	.429
Fusion-output	**.782**	**.802**	.787
Average 1-4	.760	.784	.800
Average 1-5	.774	.797	**.822**
Average 2-4	.766	.788	.798
Average 2-5	.777	.800	.814
Merged result	**.782**	**.804**	**.833**

different combination strategies are complementary, and a late merging of the averaged predictions with learned fusion-layer predictions leads to the best result. Also, when comparing "non-deep" and "deep" methods, performance of "deep" methods diminishes faster with high recall rates. This might indicate that deeply-learned features are capable of (and favor) learning the global object boundary – thus many weak edges are omitted. HED is better than other deep learning based methods in the high recall regime because deep supervision helps us to take the low level predictions into account.

Late merging to boost the average precision. We find that the weighted-fusion layer output gives best performance in F-score. However the average precision degrades compared to directly averaging all the side outputs. This might due to our focus on "global" object boundaries for the fusion-layer weight learning. Taking advantage of the readily available side outputs in HED, we merge the fusion layer output with the side outputs (at no extra cost) in order to compensate for the loss in average precision. This simple heuristic gives us the best performance across all measures that we report in Figure 5 and Table 5.

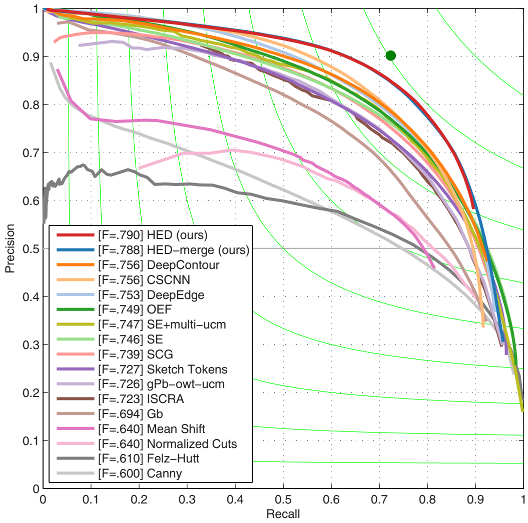

Fig. 5 Results on the BSDS500 dataset (Arbelaez et al, 2011). HED achieves the best F-score (ODS=.790, OIS=.808, AP=.811), the late-merging variant achieves best AP (ODS=.788, OIS=.808, AP=.840). Compared to several recent CNN-based edge detectors, our approach is also orders of magnitude faster. See Table 5 for detailed discussions.

Preserve aspect ratio and perform scale augmentation.

After our conference paper, we updated the training procedure for HED in two major ways. Originally, in our ICCV15 paper (Xie and Tu, 2015), we resize all the BSDS500 training and test images to a fixed size of 400 × 400. We call this HED (400 × 400), as seen in Table 5. In our updated approach, we keep the image aspect ratio in training and process the BSDS500 images in their input size (320 × 480 or 480 × 320) in testing, which makes HED more general and not dependent on the input image size. We also enhance data augmentation by scaling the training images to 50%, 100%, and 150% of the original BSDS500 images, resulting in triple the training data compared to (Xie and Tu, 2015). With the changes listed above and under the same network structure, we are able to improve the F-score from 0.782 by the original HED to a new result, 0.790 by the updated HED.

Figure 5 shows the precision-recall of the previous edge detection methods and HED on the BSDS500 dataset; Table 5 shows a detailed quantitative measures between these competing approaches. The first group in Figure 7 shows some qualitative results on BSDS500 images where the first three examples receive relatively high F-scores and the last example (lower-right) produces a relatively low F-score.

Table 5 Results on BSDS500 (Arbelaez et al, 2011). ∗ refers to results on the BSDS300 datset in (Martin et al, 2004) and † indicates GPU time. HED-fusion (DSN) (400 × 400) refers to the HED algorithm reported in (Xie and Tu, 2015) that resizes for all the images in the BSDS500 dataset to a fixed size of 400 × 400.

	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.600	.640	.580	15
Felz-Hutt (Felzenszwalb and Huttenlocher, 2004)	.610	.640	.560	10
BEL (Dollár et al, 2006)	.660∗	-	-	1/10
gPb-owt-ucm (Arbelaez et al, 2011)	.726	.757	.696	1/240
Sketch Tokens (Lim et al, 2013)	.727	.746	.780	1
SCG (Ren and Bo, 2012)	.739	.758	.773	1/280
SE-Var (Dollár and Zitnick, 2015)	.746	.767	.803	2.5
OEF (Hallman and Fowlkes, 2014)	.749	.772	.817	-
DeepNets (Kivinen et al, 2014)	.738	.759	.758	1/5†
N4-Fields (Ganin and Lempitsky, 2014)	.753	.769	.784	1/6†
DeepEdge (Bertasius et al, 2015)	.753	.772	.807	$1/10^3$†
CSCNN (Hwang and Liu, 2015)	.756	.775	.798	-
DeepContour (Shen et al, 2015)	.756	.773	.797	1/30†
HED-fusion (DSN) (400 × 400) (Xie and Tu, 2015)	.782	.804	.833	2.5†
HED-fusion (DSN)	**.790**	**.808**	.811	2.5†
HED-fusion (no DSN)	.785	.801	.730	2.5†
HED-late-merging	.788	.808	**.840**	2.5†

4.3 NYUD Dataset

The NYU Depth (NYUD) dataset (Silberman et al, 2012) has 1449 RGB-D images. This dataset was used for edge detection in (Ren and Bo, 2012) and (Gupta et al, 2013). Here we use the setting described in (Dollár and Zitnick, 2015) and evaluate HED on data processed by (Gupta et al, 2013). The NYUD dataset is split into 381 training, 414 validation, and 654 testing images. All images are made to the same size and we train our network on full resolution images. As used in (Gupta et al, 2014; Dollár and Zitnick, 2015), during evaluation we increase the maximum tolerance allowed for correct matches of edge predictions to ground truth from 0.0075 to 0.011.

Depth information encoding. Following the success in (Gupta et al, 2014) and (Long et al, 2015), we leverage the depth information by utilizing HHA features in which the depth information is embedded into three channels: horizontal disparity, height above ground, and angle of the local surface normal with the inferred direction of gravity . We use the same HED architecture and hyper-parameter settings as were used for BSDS500. We train two different models in parallel, one on RGB images and another on HHA feature images, and report the results below. We directly average the RGB and HHA predictions to produce the final result by leveraging RGB-D information. We tried other approaches to incorporate the depth information, for example, training on the raw depth channel, or concatenating the depth channel with RGB channels before the first convolutional layer, however, none of these attempts led to notable improvements compared using HHA. The effectiveness of the

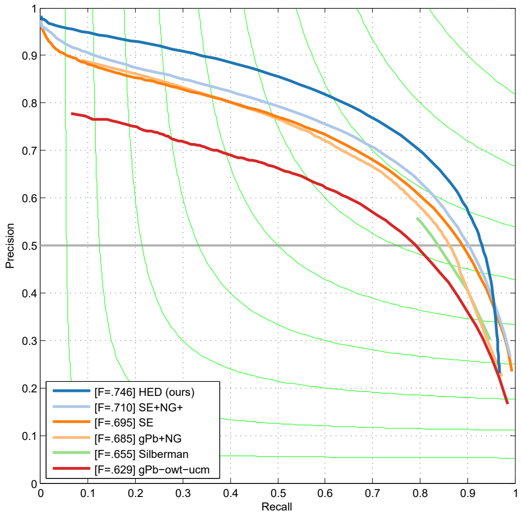

Fig. 6 Precision/recall curves on NYUD dataset (Silberman et al, 2012). Holistically-nested edge detection (HED) trained with RGB and HHA features achieves the best result (ODS=.746). See Table 6 for additional information.

HHA features shows that, although deep neural networks are capable of automatic feature learning, for depth data, carefully hand-designed features are still necessary, especially when training data is limited.

Table 6 Results on the NYUD dataset (Silberman et al, 2012) †GPU time

	ODS	OIS	AP	FPS
gPb-ucm	.632	.661	.562	1/360
Silberman (Silberman et al, 2012)	.658	.661	-	<1/360
gPb+NG(Gupta et al, 2013)	.687	.716	.629	1/375
SE(Dollár and Zitnick, 2015)	.685	.699	.679	5
SE+NG+(Gupta et al, 2014)	.710	.723	.738	1/15
HED-RGB	.720	.734	.734	2.5†
HED-HHA	.682	.695	.702	2.5†
HED-RGB-HHA	**.746**	**.761**	**.786**	1†

Table 6 and Figure 6 show the precision-recall evaluations of HED in comparison to other methods. Our training procedures are the same as for BSDS500. During testing, we use the *Average 2-4* prediction instead of the Fusion-layer output as *Average 2-4* yields top performance. We do not perform late merging since combining two sources of edge map predictions (RGB and HHA) already gives good average precision. Note that the performance using only RGB values already exceed those of previous approaches. The second group in Figure 7 shows some qualitative results on NYUD images where the first three examples receive relatively high F-scores and the last example (lower-right) produces a relatively low F-score.

4.4 Edges vs. Boundaries: Multicue Dataset

A Multicue edge and boundary dataset, motivated from psychophysics research, is recently proposed in (Mély et al, 2015), which consists of 100 short binocular video clips of natural scenes captured by a stereo camera. For each video clip (10 frames each), only the last frame of the left sequence is manually labeled by human subjects. We call this dataset "Multicue". There are several notable differences between the Multicue dataset and the BSDS500 dataset: (1) Multicue captures complex scenes whereas BSDS500 contains object-centric images of relatively lower complexity. (2) Each frame in Multicue is of size $1,280 \times 720$ which is much higher than the 480×320 size in BSDS500. (3) Two sets of ground-truth annotations are obtained in Multicue: one accounting for object boundaries and another focusing on low-level edges. These two annotations capture different levels of visual perception on the same set of images and enable us to study interesting properties of edge detectors that are not observable when experimenting on the BSDS500 dataset.

In computer vision, "object boundaries" and "edges" are sometimes used interchangeably, however under their strict definitions, they refer to visual perception at different stages. Edges are modeled in the early stages of the perception system, e.g. V1 and boundaries correspond to high-level semantics that are modeled in the later stages of the perception system, e.g. V4. Note that in this part of the experiment, we follow the definition of "edges" and "boundaries" as defined in (Mély et al, 2015), which is somewhat different from the "edge" definition in the previous sections when referring to the BSDS500 dataset. One might expect deep learning based edge detectors to excel when trained to detect (high-level) "boundaries", and to be less effective when trained to detect (low-level) "edges". Here, we evaluate the performance of HED by experimenting on the Multicue dataset.

We keep the same algorithmic and network settings for HED when training on BSDS500 and on Multicue. Since the resolution of Multicue is much higher, we randomly crop 500×500 sub-images during training. To further alleviate the scarcity of the labeled data, we augment the training set by random horizontal flipping, rotating (90, 180, and 270 degrees) and scaling (75% and 125%) images and their corresponding ground-truth label maps. We use a learning rate of 1e-6, weight decay 0.0002 and train the model for 2,000 iterations. Following (Mély et al, 2015), we randomly split the dataset into 80 training and 20 testing images, and report the averaged score over three independent runs.

Table 7 shows a comparison between HED (trained separately on boundaries and edges), the method reported in (Mély et al, 2015), and human subjects. In (Mély et al, 2015), the authors explicitly study multiple image cues including intensity, luminance, single/double-opponent color, motion and stereo,

Table 7 Edge and boundary detection results on the Multicue dataset (Mély et al, 2015).

	ODS	OIS	AP
Human-Boundary	.760 (0.017)	-	-
Multicue-Boundary (Mély et al, 2015)	.720 (0.014)	-	-
HED-Boundary	**.814** (0.011)	.822 (0.008)	.869 (0.015)
Human-Edge	.750 (0.024)	-	-
Multicue-Edge (Mély et al, 2015)	.830 (0.002)	-	-
HED-Edge	**.851**(0.014)	.864 (0.011)	.890 (0.007)

followed by a fusion procedure to form a region-based mid-level representation, which is then used for training a $\mathcal{L}_2$-norm regularized logistic classifier. In "boundary" detection, HED outperforms the method in (Mély et al, 2015) by a large margin of 9.4% absolute improvement. In the task of "edge" detection, HED wins by a relatively smaller gap of 2.1%. These observations are understandable and consistent with our hypothesis: for a low-level vision task, features such as color and luminance already provide informative cues; for a high-level vision task, multi-scale and multi-level feature learning plays a more important role.

Qualitative results on some Multicue images are shown in the third group (Multicue-edge) and the fourth group (Multicue-boundary) in Figure 7, where each group consists of three example images receiving relatively high F-scores and one example (lower-right) observing a relatively low F-score.

4.5 From Segmentation to Edge Detection: PASCAL-Context Dataset

In this section, we validate HED on another widely used computer vision benchmark, PASCAL-**Context**(Mottaghi et al, 2014), which is an extension to the PASCAL VOC-2010 image segmentation dataset (Everingham et al, 2014), in which 11,530 images are composed of a wide variety of object categories beyond the original 20 object classes in VOC 2010. Recent work (Yang et al, 2016; Maninis et al, 2016) shows edge detection results on the original PASCAL VOC dataset. We argue that the PASCAL-Context dataset might be more suitable for our study in this paper, as it is fully labeled, thus has richer and more diverse edge labelings.

Each image in the PASCAL-Context dataset is associated with a ground-truth label map where each pixel is assigned with a semantic label; generating edges from image labeling is straight-forward: a pixel is considered a boundary pixel if any of its neighbors has a different label. In this experiment, we use the 60-category version of the dataset. We use the train and validation split provided in the original dataset. We train HED on the train split (4998 images) and report results on the validation split (5105 images). We keep the same settings as we use in the BSDS500 experiment, except reducing the initial learning rate to 1e-7. Similar to the NYUD dataset, due to the inconsistency of the annotations, we increase the matching tolerance to 0.011 while evaluating on PASCAL-Context. We show the cross-dataset evaluation results with BSDS-trained model and PASCAL-Context trained model in Table 8. Inspired by the cross-dataset validation in (Zhu et al, 2015; Premachandran et al, 2015), we investigate how HED trained on BSDS500 generalizes for detecting boundaries on PASCAL-Context and vice versa. For the *PASCAL-Context on PASCAL-Context* experiment, we train the model for 80k iterations. For the *PASCAL-Context on BSDS500* experiment, the results are evaluated with a model trained for 5k iterations to avoid severe over-fitting (in a sense of cross-dataset generalization) to the strong object boundaries in PASCAL-Context.

Table 8 Cross-dataset results with the PASCAL-Context dataset and the BSDS500 dataset (Arbelaez et al, 2011).

	ODS	OIS	AP
BSDS500 on PASCAL-Context	.526	.552	.397
PASCAL-Context on PASCAL-Context	.584	.592	.443
PASCAL-Context on BSDS500	.778	.795	.814
BSDS500 on BSDS500	.790	.808	.811

We report the results in Table 8. Edge detection results on PASCAL-Context dataset are generally worse than those on BSDS500. This shows PASCAL-Context is a more challenging dataset as an edge detection benchmark, partially due to the inconsistent and noisy annotations. HED model trained on PASCAL-Context works surprisingly well on BSDS500 (ODS score 0.778), which suggests that features learned on semantic segmentation datasets can generalize well to the general boundary detection task. In Table 8, we also observe that training HED on the target dataset is consistently better than cross-dataset validation, which has been previously reported in (Zhu et al, 2015). Qualitative results on some PASCAL-Context images are shown in the fifth group in Figure 7.

5 Conclusion

In this paper, we have developed a new convolutional-neural-network-based edge detection system that demonstrates state-of-the-art performance on natural images at a speed of practical relevance (e.g., 0.4 seconds using GPU and 12 seconds using CPU). Our algorithm builds on top of the ideas of fully convolutional neural networks and deeply-supervised nets. We also initialize our network structure and parameters by adopting a pre-trained trimmed VGGNet. Our method shows the state-of-the-art results in edge/boundary detection on widely adopted benchmark datasets by combining multi-scale and multi-level visual responses, even though explicit contextual and high-level information has not been enforced. Source code and pretrained models are available online at `https://github.com/s9xie/hed`.

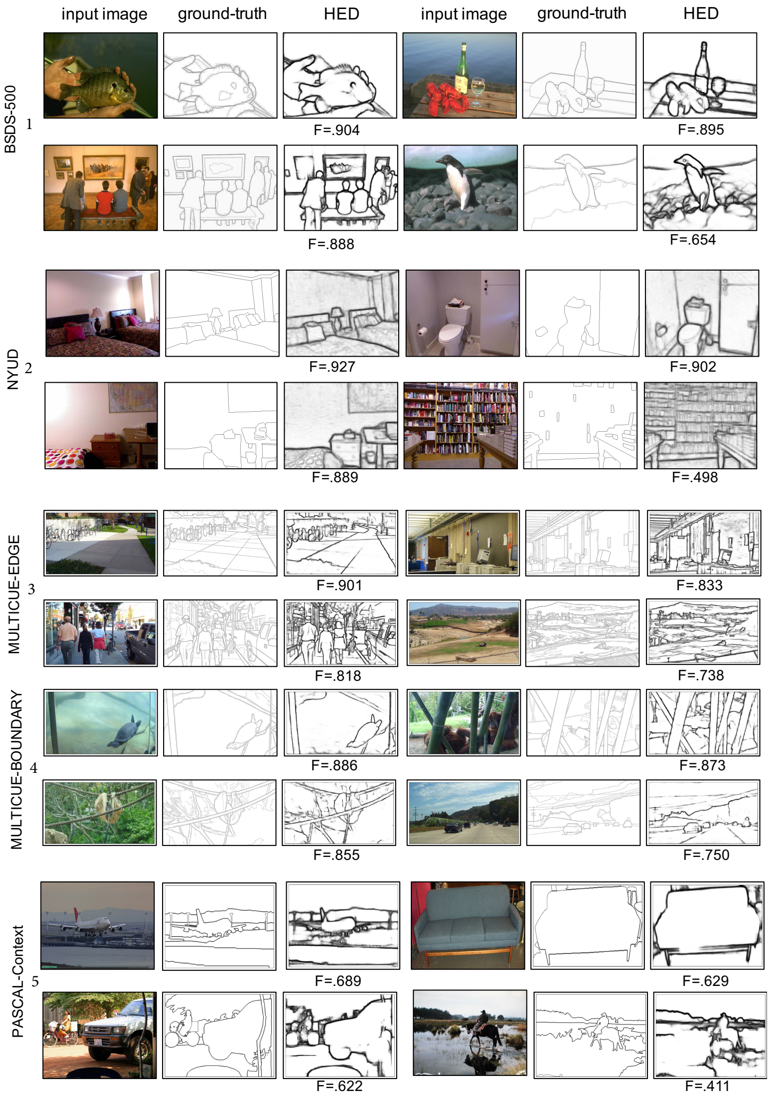

Fig. 7 HED results on four benchmark datasets including BSDS500 (Arbelaez et al, 2011), NYUD (Silberman et al, 2012), Multicue-edge/boundary (Mély et al, 2015), and PASCAL-Context (Everingham et al, 2014). Results are presented in five groups; the first three results in each group are selected from top performing examples (relatively high F-scores) and the last one (lower-right) shows an under performing example (a relatively low F-score).

Acknowledgment. This work is supported by NSF NSF IIS-1618477IIS-1216528 (IIS-1360566), NSF award IIS-0844566 (IIS-1360568), NSF IIS-1618477, and a Northrop Grumman Contextual Robotics grant. We thank Patrick Gallagher and Jameson Merkow for helping improve this manuscript. We also thank Piotr Dollár and Yin Li for insightful discussions. We are grateful for the generous donation of the GPUs by NVIDIA.

References

Arbelaez P, Maire M, Fowlkes C, Malik J (2011) Contour detection and hierarchical image segmentation. PAMI 33(5):898–916

Bertasius G, Shi J, Torresani L (2015) Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In: CVPR

Buyssens P, Elmoataz A, Lézoray O (2013) Multiscale convolutional neural networks for vision–based classification of cells. In: ACCV

Canny J (1986) A computational approach to edge detection. PAMI (6):679–698

Chen LC, Barron JT, Papandreou G, Murphy K, Yuille AL (2015) Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. arXiv preprint arXiv:151103328

Dollár P, Zitnick CL (2015) Fast edge detection using structured forests. PAMI

Dollár P, Tu Z, Belongie S (2006) Supervised learning of edges and object boundaries. In: CVPR

Elder JH, Goldberg RM (2002) Ecological statistics of gestalt laws for the perceptual organization of contours. Journal of Vision 2(4):5

Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J, Zisserman A (2014) The pascal visual object classes challenge: A retrospective. IJCV 111(1):98–136

Farabet C, Couprie C, Najman L, LeCun Y (2013) Learning hierarchical features for scene labeling. PAMI

Felzenszwalb PF, Huttenlocher DP (2004) Efficient graph-based image segmentation. IJCV 59(2):167–181

Ganin Y, Lempitsky V (2014) N4-fields: Neural network nearest neighbor fields for image transforms. arXiv preprint arXiv:14066558

Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, pp 580–587

Gupta S, Arbelaez P, Malik J (2013) Perceptual organization and recognition of indoor scenes from rgb-d images. In: CVPR

Gupta S, Girshick R, Arbeláez P, Malik J (2014) Learning rich features from rgb-d images for object detection and segmentation. In: ECCV

Hallman S, Fowlkes CC (2014) Oriented edge forests for boundary detection. arXiv preprint arXiv:14124181

Hariharan B, Arbeláez P, Girshick R, Malik J (2015) Hypercolumns for object segmentation and fine-grained localization. In: CVPR

Hoiem D, Stein AN, Efros AA, Hebert M (2007) Recovering occlusion boundaries from a single image. In: ICCV

Hoiem D, Efros AA, Hebert M (2008) Putting objects in perspective. IJCV 80(1):3–15

Hou X, Yuille A, Koch C (2013) Boundary detection benchmarking: Beyond f-measures. In: CVPR

Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology 160(1):106–154

Hwang JJ, Liu TL (2015) Pixel-wise deep learning for contour detection. In: ICLR

Khoreva A, Benenson R, Omran M, Hein M, Schiele B (2016) Weakly supervised object boundaries. In: CVPR

Kittler J (1983) On the accuracy of the sobel edge detector. Image and Vision Computing 1(1):37–42

Kivinen JJ, Williams CK, Heess N, Technologies D (2014) Visual boundary prediction: A deep neural prediction network and quality dissection. In: AISTATS

Kokkinos I (2016) Pushing the boundaries of boundary detection using deep learning. In: ICLR

Konishi S, Yuille AL, Coughlan JM, Zhu SC (2003) Statistical edge detection: Learning and evaluating edge cues. PAMI 25(1):57–74

LeCun Y, Boser B, Denker JS, Henderson D, Howard R, Hubbard W, Jackel L (1989) Backpropagation applied to handwritten zip code recognition. In: Neural Computation

Lee CY, Xie S, Gallagher P, Zhang Z, Tu Z (2015) Deeply-supervised nets. In: AISTATS

Li Y, Paluri M, Rehg JM, Dollár P (2016) Unsupervised learning of edges. In: CVPR

Lim JJ, Zitnick CL, Dollár P (2013) Sketch tokens: A learned mid-level representation for contour and object detection. In: CVPR

Liu C, Yuen J, Torralba A (2011) Nonparametric scene parsing via label transfer. PAMI 33(12):2368–2382

Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: CVPR

Maninis KK, Pont-Tuset J, Arbeláez P, Van Gool L (2016) Convolutional oriented boundaries. ECCV

Marr D, Hildreth E (1980) Theory of edge detection. Proceedings of the Royal Society of London Series B Biological Sciences 207(1167):187–217

Martin DR, Fowlkes CC, Malik J (2004) Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI 26(5):530–549

Mély D, Kim J, McGill M, Guo Y, Serre T (2015) A systematic comparison between visual cues for boundary detection. Vision research 120:93–107

Merkow J, Kriegman D, Marsden A, Tu Z (2016) Dense volume-to-volume vascular boundary detection. In: MICCAI

Mottaghi R, Chen X, Liu X, Cho NG, Lee SW, Fidler S, Urtasun R, Yuille A (2014) The role of context for object detection and semantic segmentation in the wild. In: CVPR

Neverova N, Wolf C, Taylor GW, Nebout F (2014) Multi-scale deep learning for gesture detection and localization. In: ECCV Workshops

Premachandran V, Bonev B, Yuille AL (2015) Pascal boundaries: A class-agnostic semantic boundary dataset. arXiv preprint arXiv:151107951

Ren X (2008) Multi-scale improves boundary detection in natural images. In: ECCV

Ren X, Bo L (2012) Discriminatively trained sparse code gradients for contour detection. In: NIPS

Ruderman DL, Bialek W (1994) Statistics of natural images: Scaling in the woods. Physical review letters 73(6):814

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2014) Imagenet large scale visual recognition challenge. arXiv:1409.0575

Sermanet P, Chintala S, LeCun Y (2012) Convolutional neural networks applied to house numbers digit classification. In: ICPR

Shen W, Wang X, Wang Y, Bai X, Zhang Z (2015) Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection draft version. In: CVPR

Shen W, Zhao K, Jiang Y, Wang Y, Zhang Z, Bai X (2016) Object skeleton extraction in natural images by fusing scale-associated deep side outputs. In: CVPR

Silberman N, Hoiem D, Kohli P, Fergus R (2012) Indoor segmentation and support inference from rgbd images. In: ECCV

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR

Torre V, Poggio TA (1986) On edge detection. PAMI (2):147–163

Tu Z (2008) Auto-context and its application to high-level vision tasks. In: CVPR

Van Essen DC, Gallant JL (1994) Neural mechanisms of form and motion processing in the primate visual system. Neuron 13(1):1–10

Witkin AP (1984) Scale-space filtering: A new approach to multi-scale description. In: ICASSP

Xie S, Tu Z (2015) Holistically-nested edge detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp 1395–1403

Yang J, Price B, Cohen S, Lee H, Yang MH (2016) Object contour detection with a fully convolutional encoder-decoder network. CVPR

Yuille AL, Poggio TA (1986) Scaling theorems for zero crossings. PAMI (1):15–25

Zhu Y, Tian Y, Mexatas D, Dollár P (2015) Semantic amodal segmentation. arXiv preprint arXiv:150901329