arXiv:1504.06375v2 [cs.CV] 4 Oct 2015

Holistically-Nested Edge Detection

Saining Xie
Dept. of CSE and Dept. of CogSci
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093
s9xie@eng.ucsd.edu

Zhuowen Tu
Dept. of CogSci and Dept. of CSE
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093
ztu@ucsd.edu

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 BSD500 dataset (ODS F-score of .782*) and the NYU Depth dataset (ODS F-score of* .746*), and do so with an improved speed (*0.4*s per image) that is orders of magnitude faster than some recent CNN-based edge detection algorithms.*

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 edges in natural images involves visual perception of various "levels" [18, 27]. A relatively comprehensive data collection and cognitive study [28] shows that while different subjects do have somewhat different preferences regarding where to place the edges and boundaries, there was nonetheless impressive consistency between subjects, e.g. reaching F-score 0.80 in the consistency study [28].

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 [20], zero-crossing [27, 37], and the widely adopted Canny detector [4]; methods driven by II: *information theory* on top of features arrived at through careful manual design, such as Statistical Edges [22], Pb [28], and gPb [1]; and III: *learning-based* methods that remain reliant on features of human design, such as BEL [5], Multi-scale [30], Sketch Tokens [24], and Structured Edges [6]. 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 [10], DeepContour [34], DeepEdge [2], and CSCNN [19]. Prior to this explosive development in deep learning, the Structured Edges method (typically abbreviated SE) [6] emerged as one of the most celebrated systems for edge detection, thanks to its state-of-the-art performance on the BSD500

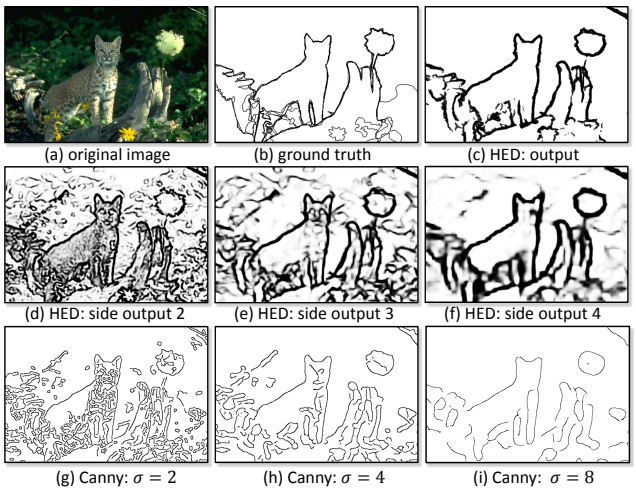

Figure 1. Illustration of the proposed HED algorithm. In the first row: (a) shows an example test image in the BSD500 dataset [28]; (b) shows its corresponding edges as annotated by human subjects; (c) displays the HED results. In the second row: (d), (e), and (f), respectively, show side edge responses 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 [4] at the scales $\sigma = 2.0$, $\sigma = 4.0$, and $\sigma = 8.0$. HED shows a clear advantage in consistency over Canny.

Dental Monitoring Ex. 2042
Align Technology, Inc., v. Dental Monitoring SAS.
IPR2023-01369
Page 1 of 10

dataset [28] (with, e.g., F-score of .746) and its practically significant speed of 2.5 frames per second. Recent CNN-based methods [10, 34, 2, 19] have demonstrated promising F-score performance improvements over SE. However, there still remains large room for improvement in these CNN-based methods, in both F-score performance and in speed — at present, time to make a prediction ranges from several seconds [10] to a few hours [2] (even when using modern GPUs).

Here, 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 [40, 41, 30] in which scale-space edge fields are neither automatically learned nor hierarchically connected. 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 [4] with different scale parameters. Not only are Canny edges at different scales not directly connected, they also exhibit spatial shift and inconsistency.

The proposed holistically-nested edge detector (HED) tackles two critical issues: (1) holistic image training and prediction, inspired by fully convolutional neural networks [26], 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 [23], that performs deep layer supervision to "guide" early classification results. We find that the favorable characteristics of these underlying techniques manifest in HED being both accurate and computationally efficient.

2. Holistically-Nested Edge Detection

In this section, we describe in detail the formulation of our proposed edge detection system. We start by discussing related neural-network-based approaches, particularly those that emphasize multi-scale and multi-level feature learning. 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 [28, 5], (2) multi-scale response fusion [40, 32, 30], (3) engagement of different levels of visual perception [18, 27, 39, 17] such as mid-level Gestalt law information [7], (4) incorporating structural information (intrinsic correlation carried within the input data and output solution) [6] and context (both short- and long- range interactions) [38], (5) making holistic image predictions (referring to approaches that perform prediction by taking the image contents globally and directly) [25], (6) exploiting 3D geometry [15], and (7) addressing occlusion boundaries [16].

Structured Edges (SE) [6] 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 [10, 34, 2, 19] 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 [2, 19] 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 [26], but with additional deep supervision on top of trimmed VGG nets [36] (shown in Figure 3). In the absence of deep supervision and side outputs, a fully convolutional network [26] (FCN) produces a less satisfactory result (e.g. F-score .745 on BSD500) 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 4); (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 3).

2.1. Existing multi-scale and multi-level NN

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"

Dental Monitoring Ex. 2042
Page 2 of 10

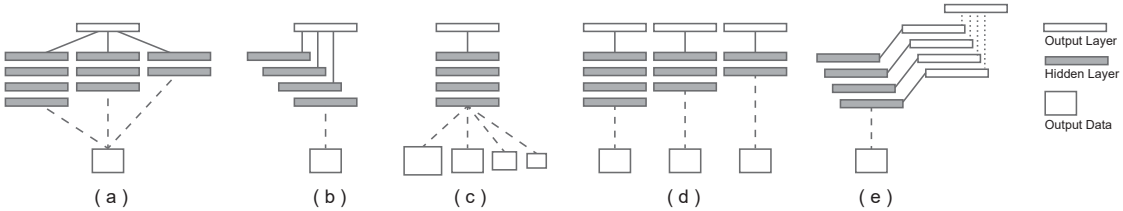

Figure 2. Illustration of different multi-scale deep learning architecture configurations: (a) multi-stream architecture; (b) skip-layer net architecture; (c) a single model running on multi-scale inputs; (d) separate training of different networks; (e) our proposed holistically-nested architectures,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

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 [3, 29] 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 [26, 14, 2, 33, 10]. 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 [8]. This approach is also common in non-deep-learning based methods [6]. Note that ensemble testing impairs the prediction efficiency of learning systems, especially with deeper models[2, 10].

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[23] 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.

2.2. Formulation

Here we formulate our approach for edge prediction. **Training Phase** We denote our input training data set by $S = \{(X_n, Y_n), n = 1, \ldots, N\}$, where sample $X_n = \{x_j^{(n)}, j = 1, \ldots, |X_n|\}$ denotes the raw input image and $Y_n = \{y_j^{(n)}, j = 1, \ldots, |X_n|\}, y_j^{(n)} \in \{0, 1\}$ 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

Dental Monitoring Ex. 2042
Page 3 of 10

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 $\mathbf{w} = (\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(M)})$. We consider the objective function

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

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 $X = (x_j, j = 1, \ldots, |X|)$ and edge map $Y = (y_j, j = 1, \ldots, |X|), y_j \in \{0, 1\}$. 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 [19], 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)

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

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 $\hat{A}_{\text{side}}^{(m)} \equiv \{a_j^{(m)}, j = 1, \ldots, |Y|\}$ 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

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

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:

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

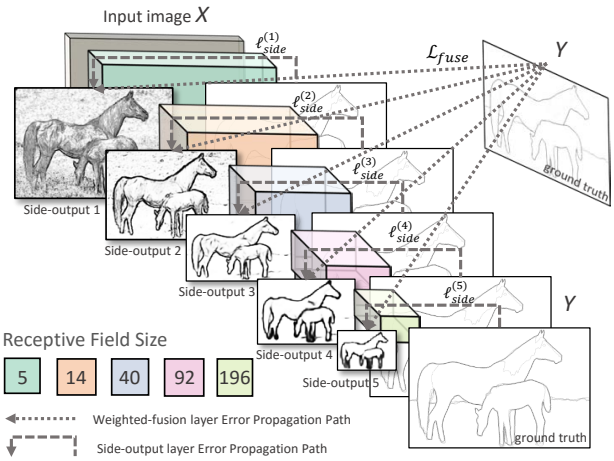

Figure 3. Illustration of our network architecture for edge detection, highlighting the error backpropagation paths. Side-output layers are inserted after convolutional layers. Deep supervision is imposed at each side-output layer, guiding the side-outputs towards edge predictions with the characteristics 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).

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

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

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

where $\text{CNN}(\cdot)$ 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.

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

3. Network Architecture

Next, we describe the network architecture of HED.

3.1. 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 [36] 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 down-sampling layers). Recent work [2] also demonstrates that

Dental Monitoring Ex. 2042
Page 4 of 10

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.

Table 1. The receptive field and stride size in VGGNet [36] 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

3.2. 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.

Table 2. Performance of alternative architectures on BSDS 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

FCN and skip-layer architecture 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

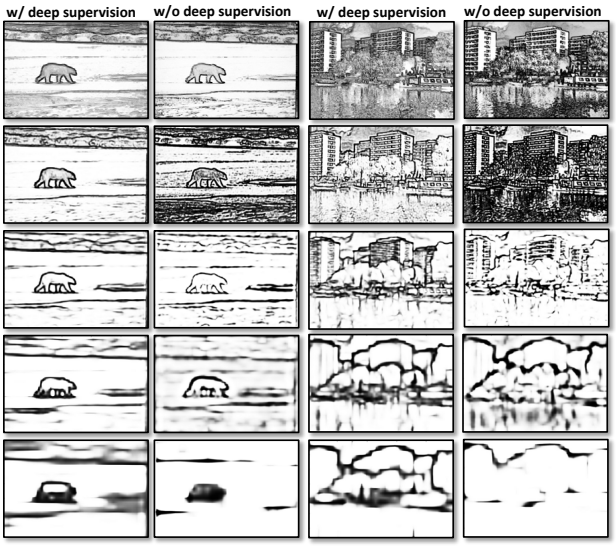

Figure 4. Two examples illustrating how deep supervision helps side-output layers to produce multi-scale dense predictions. Note that in the left column, the side outputs become progressively coarser and more "global", while critical object boundaries are preserved. In the right column, the predictions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

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 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 suc-

Dental Monitoring Ex. 2042
Page 5 of 10

cessive 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.

4.1. Implementation

We implement our framework using the publicly available *Caffe* Library and build on top of the publicly available implementations of FCN[26] and DSN[23]. Thus, relatively little engineering hacking is required. In our HED system, the whole network is fine-tuned from an initialization with the pre-trained VGG-16 Net model.

Model parameters In contrast to fine-tuning CNN to perform image classification or semantic segmentation, adapting CNN to perform low-level edge detection requires special care. Differences in data distribution, ground truth distribution, and loss function all contribute to difficulties in network convergence, even with the initialization of a pre-trained model. We first use a validation set and follow the evaluation strategy used in [6] to tune the deep model hyper-parameters. The hyper-parameters (and the values we choose) include: mini-batch size (10), learning rate (1e-6), loss-weight α_m for each side-output layer (1), momentum (0.9), initialization of the nested filters (0), initialization of the fusion layer weights (1/5), weight decay (0.0002), number of training iterations (10,000; divide learning rate by 10 after 5,000). We focus on the convergence behavior of the network. We observe that whenever training converges, the deviations in F-score on the validation set tend to be very small. In order to investigate whether including additional nonlinearity helps, we also consider a setting in which we add an additional layer (with 50 filters and a ReLU) before each side-output layer; we find that this worsens performance. On another note, we observe that our nested multi-scale framework is insensitive to input image scales; during our training process, we take advantage of this by resizing all the images to 400×400 to reduce GPU memory usage and to take advantage of efficient batch processing. In the experiments that follow, we fix the values of all hyper-parameters discussed above to explore the benefits of possible 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. Thus, there exists 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 even be considered as noise. This issue leads to problematic convergence behavior, even with the help of a pre-trained model. We observe that this mismatch leads to back-propagated gradients that explode at the high-level side-output layers. We therefore adjust how we make use of the ground truth labels in the BSDS dataset to combat this issue. Specifically, the ground truth labels are provided by multiple annotators and thus, implicitly, greater labeler consensus indicates stronger ground truth edges. We adopt a relatively brute-force solution: only assign a pixel a positive label if it is labeled as positive by at least three annotators; regard all other labeled pixels as negatives. This helps with the problem of gradient explosion in high level side-output layers. For low level layers, this consensus approach brings additional robustness to edge 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. 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 [2] 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 [26]. We fix all the deconvolutional layers to perform linear interpolation. Although it was pointed out in [26] 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. For a 320×480 image, it takes HED 400 ms to produce the final edge map (including the interface overhead), which is significantly faster than existing CNN-based methods [34, 2]. Some previous edge detectors also try to improve performance by the less desirable expedient of sacrificing efficiency (for example, by testing on input images from multiple scales and averaging the re-

Dental Monitoring Ex. 2042
Page 6 of 10

sults).

4.2. BSDS500 dataset

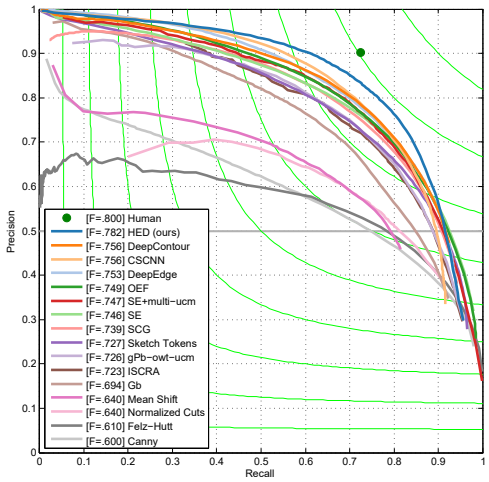

Figure 5. Results on the BSDS500 dataset. Our proposed HED framework achieves the best result (ODS=.782). Compared to several recent CNN-based edge detectors, our approach is also orders of magnitude faster. See Table 4 for a detailed discussion.

We evaluate HED on the Berkeley Segmentation Dataset and Benchmark (BSDS 500) [1] which is composed of 200 training, 100 validation, and 200 testing images. Each image has 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. The results are shown in Figure 5 and Table 4.

Table 3. Results of single and averaged side output in HED on the BSDS 500 dataset. 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**

Side outputs To explicitly validate the side outputs, we summarize the results produced by the individual side-outputs at different scales in Table 3, 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 the performance gain, either in F-score or averaged precision. To see this, in Table 3, the side-output layer 1 and layer 5 (the lowest and highest layers) achieve similar relatively low performance. One might expect these two side-output layers to not be useful in the averaged results. However this turns out not to be the case — for example, the Average 1-4 achieves ODS=.760 and incorporating the side-output layer 5, the averaged prediction achieves an ODS=.774. We find similar phenomenon when considering other ranges. As mentioned above, the predictions obtained using different combination strategies are complementary, and a late merging of the averaged predictions with learned fusion-layer predictions leads to the best result. Another observation is, when compared to previous "non-deep" methods, performance of all "deep" methods drops more in the high recall regime. This might indicate that deep 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.

Table 4. Results on BSDS500. ∗BSDS300 results,†GPU time

	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.600	.640	.580	15
Felz-Hutt [9]	.610	.640	.560	10
BEL [5]	.660∗	-	-	1/10
gPb-owt-ucm [1]	.726	.757	.696	1/240
Sketch Tokens [24]	.727	.746	.780	1
SCG [31]	.739	.758	.773	1/280
SE-Var [6]	.746	.767	.803	2.5
OEF [13]	.749	.772	.817	-
DeepNets [21]	.738	.759	.758	1/5†
N4-Fields [10]	.753	.769	.784	1/6†
DeepEdge [2]	.753	.772	.807	$1/10^3$†
CSCNN [19]	.756	.775	.798	-
DeepContour [34]	.756	.773	.797	1/30†
HED (ours)	**.782**	**.804**	**.833**	2.5†, 1/12

Late merging to boost 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

Dental Monitoring Ex. 2042
Page 7 of 10

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 4.

More training data Deep models have significantly advanced results in a variety of computer vision applications, at least in part due to the availability of large training data. In edge detection, however, we are limited by the number of training images available in the existing benchmarks. Here we want to explore whether adding more training data will help further improve the results. To do this, we expand the training set by randomly sampling 100 images from the test set. We then evaluate the result on the remaining 100 test images. We report the averaged result over 5 such trials.

We observe that by adding only 100 training images, performance improves from ODS=.782 to ODS=.797 ($\pm$.003), nearly touching the human benchmark. This shows a potentially promising direction to further enhance HED by training it with a larger dataset.

4.3. NYUDv2 Dataset

The NYU Depth (NYUD) dataset [35] has 1449 RGB-D images. This dataset was used for edge detection in [31] and [11]. Here we use the setting described in [6] and evaluate HED on data processed by [11]. 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 [12, 6], during evaluation we increase the maximum tolerance allowed for correct matches of edge predictions to ground truth from .0075 to .011.

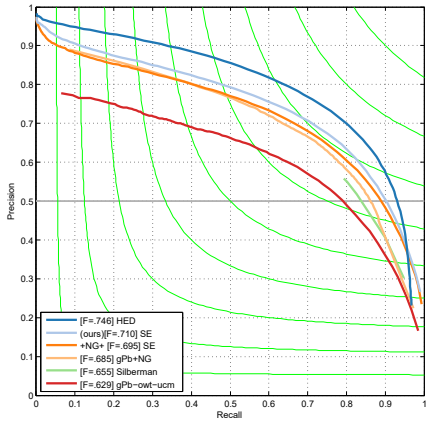

Figure 6. Precision/recall curves on NYUD dataset. Holistically-nested edge detection (HED) trained with RGB and HHA features achieves the best result (ODS=.746). See Table 5 for additional information.

Depth information encoding Following the success in [12] and [26], 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 BSDS 500. 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 also tried other approaches to incorporate the depth information, for example, by training on the raw depth channel, or by concatenating the depth channel with the RGB channels before the first convolutional layer. None of these attempts yields notable improvement compared to the approach using HHA. The effectiveness of the 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 only limited training data is available.

Table 5. Results on the NYUD dataset [35] †GPU time

	ODS	OIS	AP	FPS
gPb-ucm	.632	.661	.562	1/360
Silberman [35]	.658	.661	-	<1/360
gPb+NG[11]	.687	.716	.629	1/375
SE[6]	.685	.699	.679	5
SE+NG+[12]	.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 5 and Figure 6 show the precision-recall evaluations of HED in comparison to other competing methods. Our network structures for training are kept the same as for BSDS. During testing we use the *Average2-4* prediction instead of the Fusion-layer output as it yields the best 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 results achieved using the RGB modality only are already better than those of the previous approaches.

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 promising results in performing image-to-image learning 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.

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

A. More Results

After the ICCV submission, we retrained our model with the following : (1) In data augmentation, we further triples the dataset by scaling the training images to 50%, 100%, 150% of its original size. (2) In training phase, we use full-resolution images instead of resizing them to 400×400.

Updated results on BSDS500 benchmark dataset with this newly trained model are reported in Figure 7 and Table 6.

In the new experiment settings, while we found that the gap in F-score narrows between models with/without deep supervision, we have similar qualitative and quantitative observations as illustrated in Section 3.2.

Table 6. Updated HED results on the BSDS500 dataset.

	ODS	OIS	AP
fusion-output (with deep supervision)	**.790**	**.808**	.811
fusion-output (w/o deep supervision)	.785	.801	.730
HED (late-merging)	.788	.808	**.840**

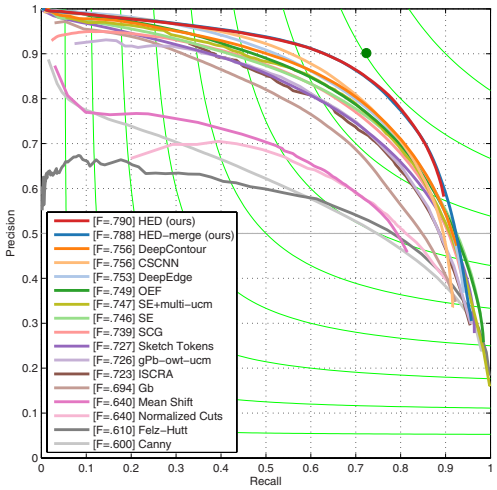

Figure 7. Updated results on the BSDS500 dataset. Our proposed HED framework achieves the best F-score (ODS=.790, OIS=.808, AP=.811), the late-merging variant achieves best average precision (ODS=.788, OIS=.808, AP=.840).

Changelog

v2 Fix typos and reorganize formulations. Add Table 2 to discuss the role of deep supervision. Add appendix A for updated results on BSDS500 in a new experiment setting. Add links to publicly available repository for training/testing code, augmented data and pre-trained model.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 1, 7

[2] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015. 1, 2, 3, 5, 6, 7

[3] P. Buyssens, A. Elmoataz, and O. Lézoray. Multiscale convolutional neural networks for vision–based classification of cells. In *ACCV*. 2013. 3

[4] J. Canny. A computational approach to edge detection. *PAMI*, (6):679–698, 1986. 1, 2

[5] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006. 1, 2, 7

[6] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 2015. 1, 2, 3, 6, 7, 8

[7] J. H. Elder and R. M. Goldberg. Ecological statistics of gestalt laws for the perceptual organization of contours. *Journal of Vision*, 2(4):5, 2002. 2

[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 2013. 3

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 7

[10] Y. Ganin and V. Lempitsky. N4-fields: Neural network nearest neighbor fields for image transforms. *arXiv preprint arXiv:1406.6558*, 2014. 1, 2, 3, 7

[11] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, 2013. 8

[12] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, 2014. 8

[13] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. *arXiv preprint arXiv:1412.4181*, 2014. 7

[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 3

[15] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 80(1):3–15, 2008. 2

[16] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007. 2

[17] X. Hou, A. Yuille, and C. Koch. Boundary detection benchmarking: Beyond f-measures. In *CVPR*, 2013. 2

[18] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962. 1, 2

[19] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. In *ICLR*, 2015. 1, 2, 4, 7

Dental Monitoring Ex. 2042
Page 9 of 10

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

[21] J. J. Kivinen, C. K. Williams, N. Heess, and D. Technologies. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, 2014. 7

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

[23] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 2, 3, 6

[24] J. J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 1, 7

[25] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *PAMI*, 33(12):2368–2382, 2011. 2

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3, 6, 8

[27] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980. 1, 2

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

[29] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout. Multi-scale deep learning for gesture detection and localization. In *ECCV Workshops*, 2014. 3

[30] X. Ren. Multi-scale improves boundary detection in natural images. In *ECCV*. 2008. 1, 2

[31] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 7, 8

[32] D. L. Ruderman and W. Bialek. Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814, 1994. 2

[33] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *ICPR*, 2012. 3

[34] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection draft version. In *CVPR*, 2015. 1, 2, 6, 7

[35] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*. 2012. 8

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 4, 5

[37] V. Torre and T. A. Poggio. On edge detection. *PAMI*, (2):147–163, 1986. 1

[38] Z. Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008. 2

[39] D. C. Van Essen and J. L. Gallant. Neural mechanisms of form and motion processing in the primate visual system. *Neuron*, 13(1):1–10, 1994. 2

[40] A. P. Witkin. Scale-space filtering: A new approach to multi-scale description. In *ICASSP*, 1984. 2

[41] A. L. Yuille and T. A. Poggio. Scaling theorems for zero crossings. *PAMI*, (1):15–25, 1986. 2

Dental Monitoring Ex. 2042
Page 10 of 10