Published in Image Processing On Line on 2022–10–03.
Submitted on 2022–09–13, accepted on 2022–09–19.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.422

A Brief Analysis of the Holistically-Nested Edge Detector

Rafael Grompone von Gioi[1], Gregory Randall[2]

[1]Centre Borelli, Université Paris-Saclay, ENS Paris-Saclay, France CMLA, (grompone@ens-paris-saclay.fr)
[2]IIE, Fing, UdelaR, Uruguay (randall@fing.edu.uy)

Communicated by J. Matías Di Martino *Demo edited by* Rafael Grompone von Gioi

Abstract

This work describes the HED method for edge detection. HED uses a neural network based on a VGG16 backbone, supplemented with some extra layers for merging the results at different scales. The training was performed on an augmented version of the BSDS500 dataset. We perform a brief analysis of the results produced by HED, highlighting its quality but also indicating its limitations. Overall, HED produces state-of-the-art results.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.txt` file of the archive. The code is a minor modification from Simon Niklaus' pytorch-hed implementation[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: image edge detection; neural network; VGG16

1 Introduction

Edge detection is a classic problem in computer vision and there is an extensive literature describing methods, metrics and comparisons [6]. In this work, we focus on the Holistically-Nested Edge Detector (HED) proposed in 2015 by Saining Xie and Zhuowen Tu [9]. HED has become very popular and inspired several other deep learning edge detectors [8, 3]. The HED method is based on a neural network and uses VGG16 [7] as a backbone. Here we briefly describe the method and then perform an analysis of its results with some experiments. For this and for the associated online demo, we use (with minor modifications) the re-implementation by Simon Niklaus [5].

[1] https://doi.org/10.5201/ipol.2022.422
[2] https://github.com/sniklaus/pytorch-hed, commit hash: 8db09037f2491abbed4e5d7e37ec75210a2dbfbf.

2 The HED Method

HED is an image edge detector based on a deep neural network. Figure 1 presents the architecture. The input is a color image (three channels) of H columns and W rows. The output is a $H \times W$ map with scores between zero and one (or equivalently between 0 and 255) indicating the confidence on the presence of an edge at each pixel position. Each pixel of the RGB input must have values in $[0, 255]$, which are normalized by subtracting the vector $(122.67891434, 116.66876762, 104.00698793)$ at each position before applying the image to the network. The trained parameters assume this normalization, which is probably related to the mean values of the training dataset.

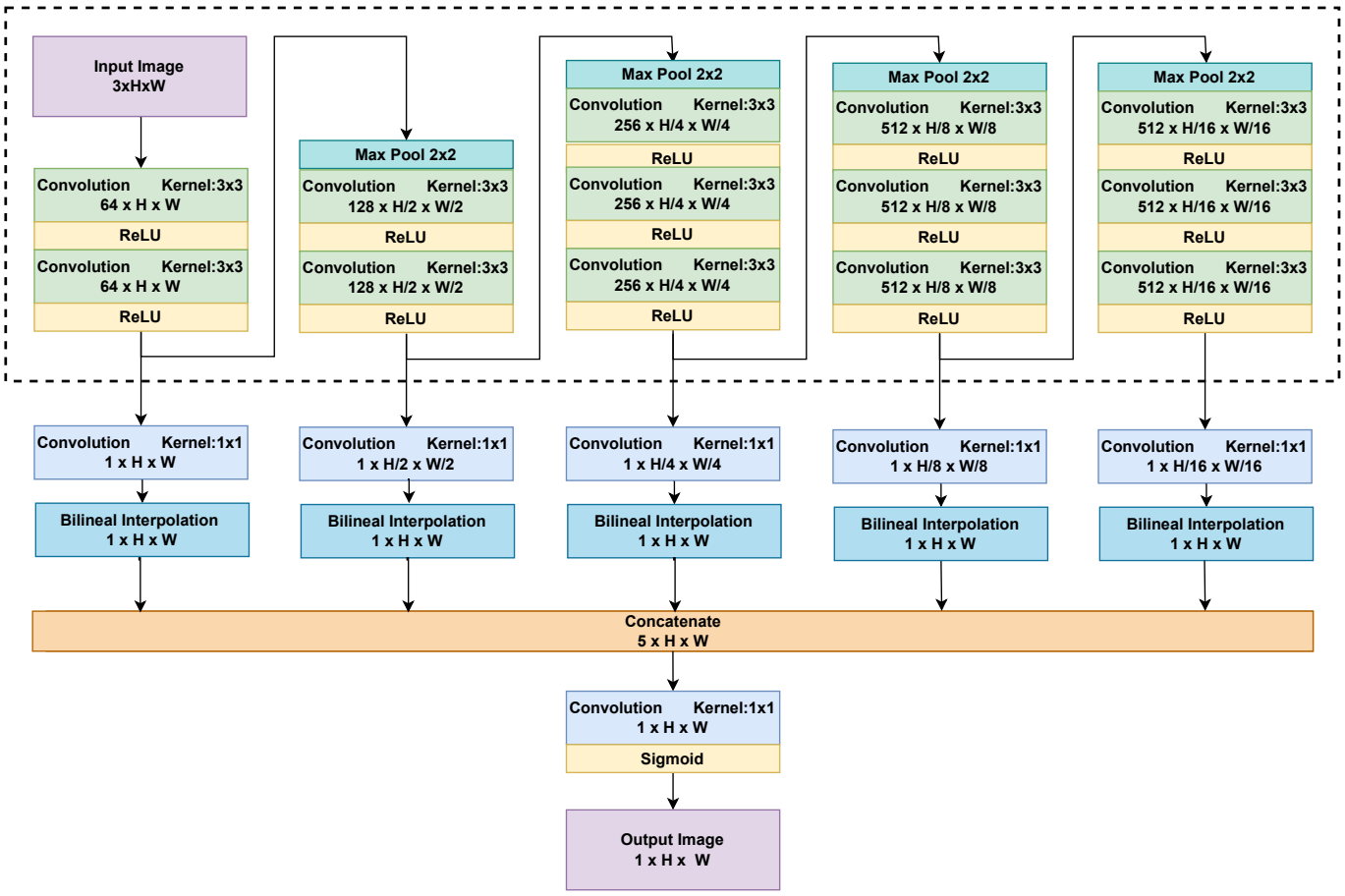


Figure 1: HED's neural network architecture. The dotted box indicates the part borrowed from VGG16 [7]. In total, the HED neural network has 14,716,171 parameters.

The HED method uses the convolutional part of the VGG16 network [7] as a backbone (encircled by a dotted box in Figure 1). This part is composed of 13 convolutional layers using 3×3 kernels, including ReLU activation functions after each of these layers. The network is organized into five groups, working at different scales: the first group works at the input image full resolution; the following groups work at half, 1/4, 1/8 and 1/16 resolution, respectively. The resolution is reduced from one group to the next by a max-pool operation with a 2×2 kernel and stride 2×2. The first group is composed of two convolutional layers with 64 filters on each one. The second group is composed of two convolutional layers with 128 filters on each one. The last three groups are composed of three convolutional layers with 256, 512 and 512 filters on each one, respectively. The rest of the layers of VGG16 are not used in HED.

The output of the five VGG16 groups (which are trained to represent edge maps at five different scales) are then merged to produce the final output of HED. To this aim, the output of each group is

processed by a 1×1 convolutional layer. This is equivalent to performing a weighed mean of the 64, 128, 256, 512 or 512 channels, respectively. The vector value at each position is reduced to a single value. Then, these partial outputs need to be interpolated to obtain again the initial resolution of the input image. This is done for each of the five resolutions using bilinear interpolation. At this stage there are five full resolution maps, which are concatenated to produce a $5 \times W \times H$ map. A final convolution with a 1×1 kernel followed by a sigmoid non-linearity is used to produce a single output value per position. Prior to the sigmoid, this is equivalent to a weighed mean of the five up-scaled partial outputs; ultimately, this is a weighted mean of the 1472 features computed at different scales.

In total, there are 19 convolutional layers, adding up to 14,716,171 parameters.

3 Training

In this work we did not try to reproduce the training stage, focusing only on the resulting method. Thus, we will not describe the training procedure. It is indeed interesting to reproduce in detail the training, but this will be left for future work. Nevertheless, we will comment on the datasets used during the training to give some context to the results obtained. The HED network was initialized with the VGG16 parameters [7] and then fine-tuned to approximate the ground truth annotations of the training dataset.

The original paper [9] indicates that HED was trained on the BSDS500 dataset [1]. BSDS500 is in turn an extension of the BSDS300 dataset [4] by the addition of 200 images for testing. HED used the 300 images included in the *train* and *val* sets common to the BSDS300 and BSDS500 datasets. It can be said that HED was trained on the full BSDS300 dataset. The training set was augmented by rotating each image by 16 different angles, and by flipping each image, see Figure 2. In addition, the same procedure was performed at full scale but also after scaling the images at half and at 1.5 resolution. This leads to a total count of $300 \times 16 \times 2 \times 3 = 28800$ training images[3].

As shown in Figure 2, the HED training set seems to derive from the annotation in BSDS500. There is, however, a point which is not completely clear for us. The original HED paper states the following:

> 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. [9]

As a result, we expected to observe as ground truth on HED's training set only the pixels common to three or more annotators. However, we failed to find traces of such operation. The HED's ground truth in Figure 3 seems to correspond to the sum of the five boundary annotations provided in BSDS500. However, it is still possible that such a threshold was imposed during the training procedure; we did not analyze that part.

4 Experiments

Figures 4, 5 and 6 show sample images and the corresponding result produced by HED. Edge maps produced by HED have a single channel and the same size as the input image. Each output pixel has a value between zero and one, but are usually normalized to values between zero and 255. The

[3]The original training dataset was initially made available by the authors at `http://vcl.ucsd.edu/hed/HED-BSDS.tar`. This URL is no longer valid, but the same dataset is now available at `http://mftp.mmcheng.net/liuyun/rcf/data/HED-BSDS.tar.gz` (1.2Gb).

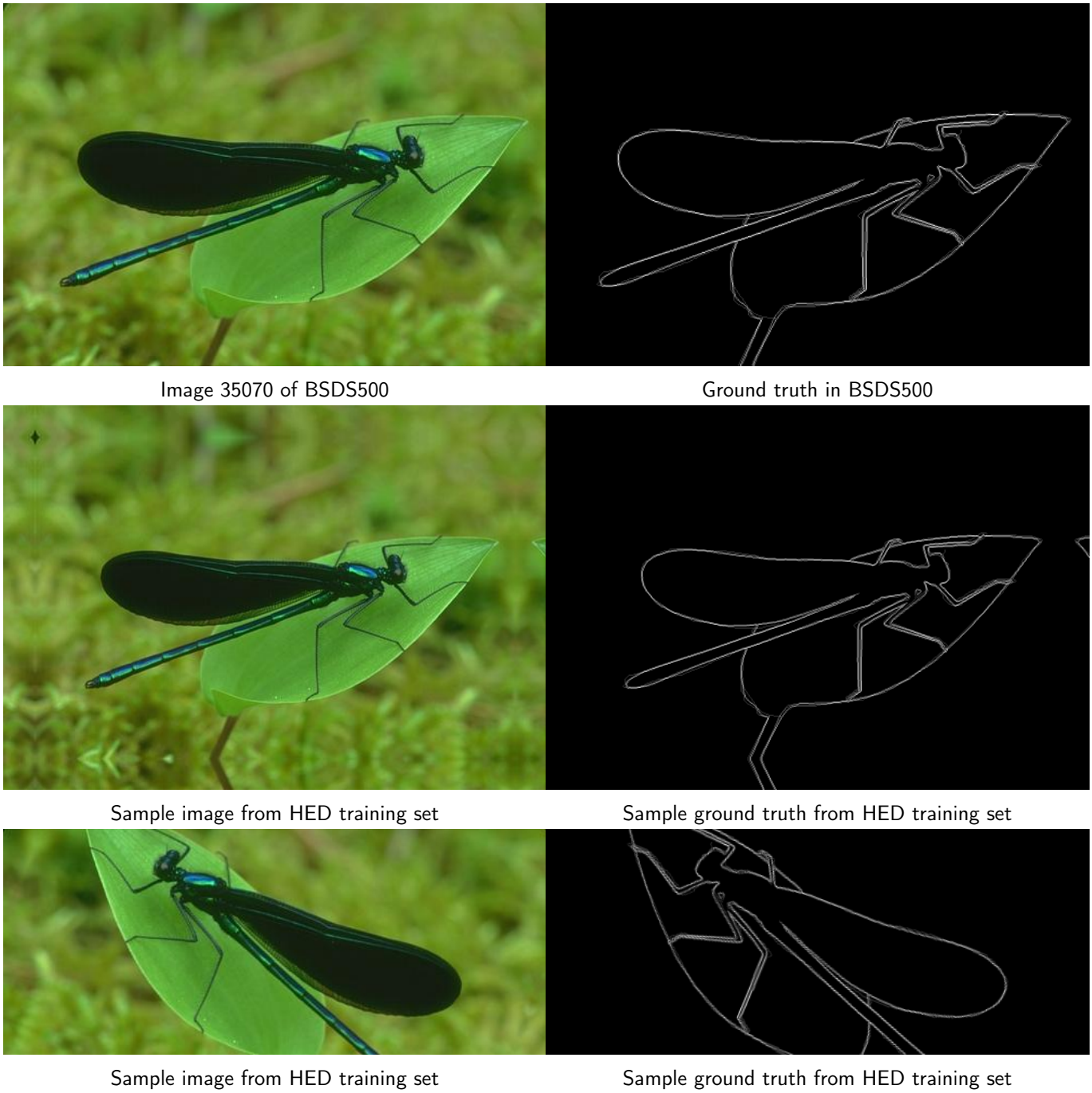


Image 35070 of BSDS500 | Ground truth in BSDS500

Sample image from HED training set | Sample ground truth from HED training set

Sample image from HED training set | Sample ground truth from HED training set

Figure 2: Example from the BSDS500 dataset [4] and two examples of data augmentation from the HED training dataset.

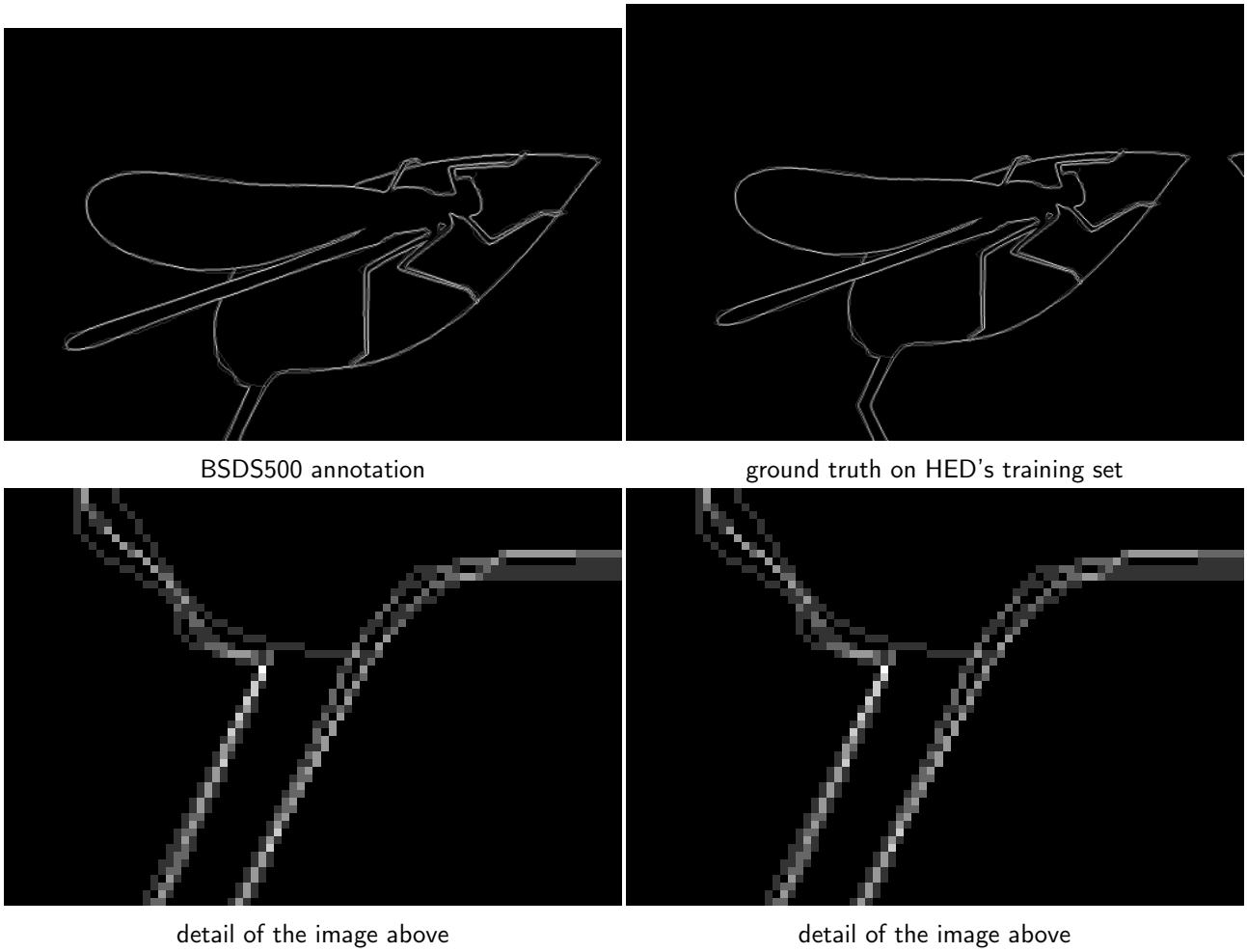


Figure 3: Left: sum of the five boundary annotations provided in BSDS500 for image 35070. Right: corresponding ground truth from HED training set. The ground truth used to train HED seems to be exactly the sum of all the annotations provided in BSDS500, normalized to the range $[0, 255]$. Note that the image sizes are not the same as a mirror-symmetric boundary condition was imposed on HED's training set. The pixel values are the same, even on parts where only one annotation is present, for example where the leaf joins the stem, see the detailed view below.

values near zero appear as black, while values near 255 appear as white. The brighter the pixel, the more confident HED is about the presence of an edge at the corresponding position.

The first example in Figure 4 is from the BSDS500 dataset. In this case the result is very good, as most of the edges in the image produce high values in the output (white), while the rest produce very low values (black). Nevertheless, this image was part of the training set, so it is not surprising that the method produces good results. It serves, anyway, to illustrate the behavior of HED in the most favorable conditions. The method is able to detect strong edges, as the contours of the wings of the dragon-fly, but also the contour of the leaf which shares almost the same shade of green as the background. The small details of the head and the legs are also well captured.

Notice that the method is able to detect edges (sharp transitions of brightness or color, as in the dragon-fly wing) as well as ridges (thin structures over a background as the legs of the dragon-fly). These two kind of structures are not differentiated; when two edges gradually approach each other (e.g. the wing gradually approaches the abdomen of the dragon-fly), the detected edges gradually merge into one detected ridge. Also, the edge map is not sharp: edges are detected as bright regions of a certain width, regardless of how sharp the edge is on the image; the regions of bright pixels corresponding to an edge are usually about five pixels wide, with slightly higher values near the center of the region.

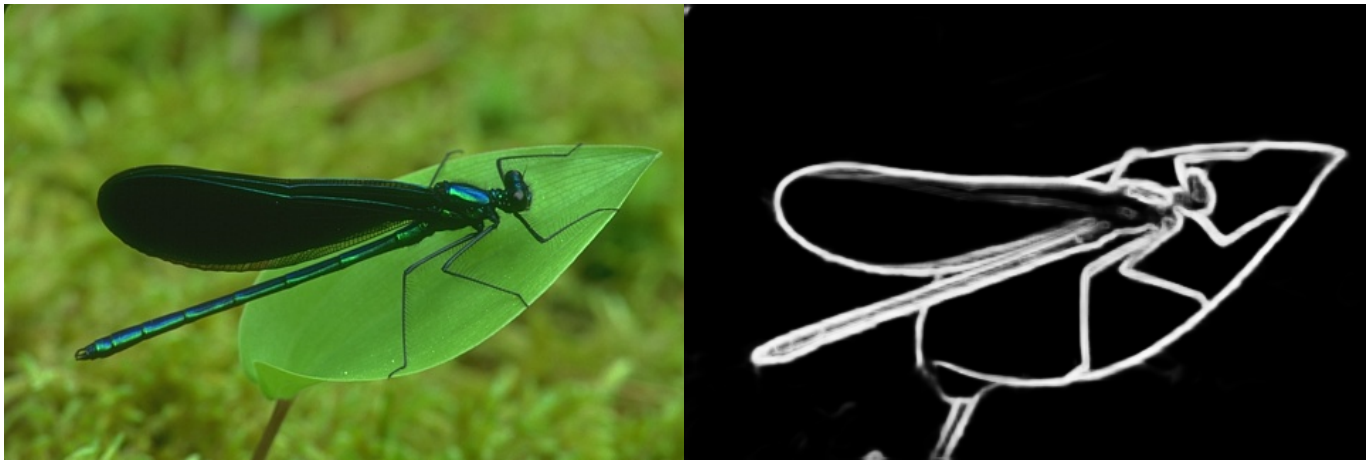


Figure 4: Input and result of HED on image number 35070 of BSDS500 dataset [1] of size 481×321.

Figure 5 shows the result of HED on a classic image for edge detection, extracted from [2]. Two versions of the image are presented: the original one, and a second version corrupted with additive Gaussian noise of $\sigma = 20$. These examples illustrate the behavior of HED on images with a different origin than the ones on which the method was trained: the image is in gray-levels instead of color, it has a different size and has a different source than the ones on the BSDS500 dataset. In spite of these differences, HED is able to produce a very good result. Most of the structures are well detected. Some details are missing, as some of the arcs to the left (probably the foot of a lamp) or the thin and long structure seen behind the head of the mannequin. As the scale of the edges grows, the detected edges become wider, as can be seen in the shadow, near the mannequin. The shadow produces a detection that gradually vanishes as the edge becomes blurred, far away from the mannequin.

On the second example of Figure 5 one can see that the addition of noise degrades the result: some of the detailed edges are no longer detected or the score is lower. The detection on the shadow also degrades. In addition, there are some partial detections on the background due to the noise.

The experiments on Figure 6 show the result of HED on the synthetic Siemens Star, a classic test pattern used to evaluate optical systems. It allows to evaluate the resolutions limits as the lines are thinner near the center; it also allows to evaluate the response at different orientations. As one can see, the edges are well detected by HED on the outer parts of the pattern. At some point, when the stripe are about 10 pixels width, the edges are no longer detected. This is probably due to the characteristics of the training set, in which medium- and high-frequency patterns appearing in textures are not annotated as edges.

A second observation is that the detection pattern is anisotropic. Note that the anisotropy appears in the shape of the central region of non detection, but also far from the center where one can see patterns presenting a weaker response. The second experiment of Figure 6 was performed with the same input image but turned 90 degrees; both types of anisotropic patterns (the shape of the non detection and the patterns of weaker response) are essentially unaltered. We can conclude that these anisotropic patterns are intrinsic to HED (and the trained parameters) and not due to artifacts of the input image.

5 Discussion

The Holistically-Nested Edge Detector (HED) [9] is a state-of-the-art neural network for edge detection which produces very good results in general. HED provides a score per pixel, indicating the confidence on the presence of an edge at each position. This is slightly different to the annotation

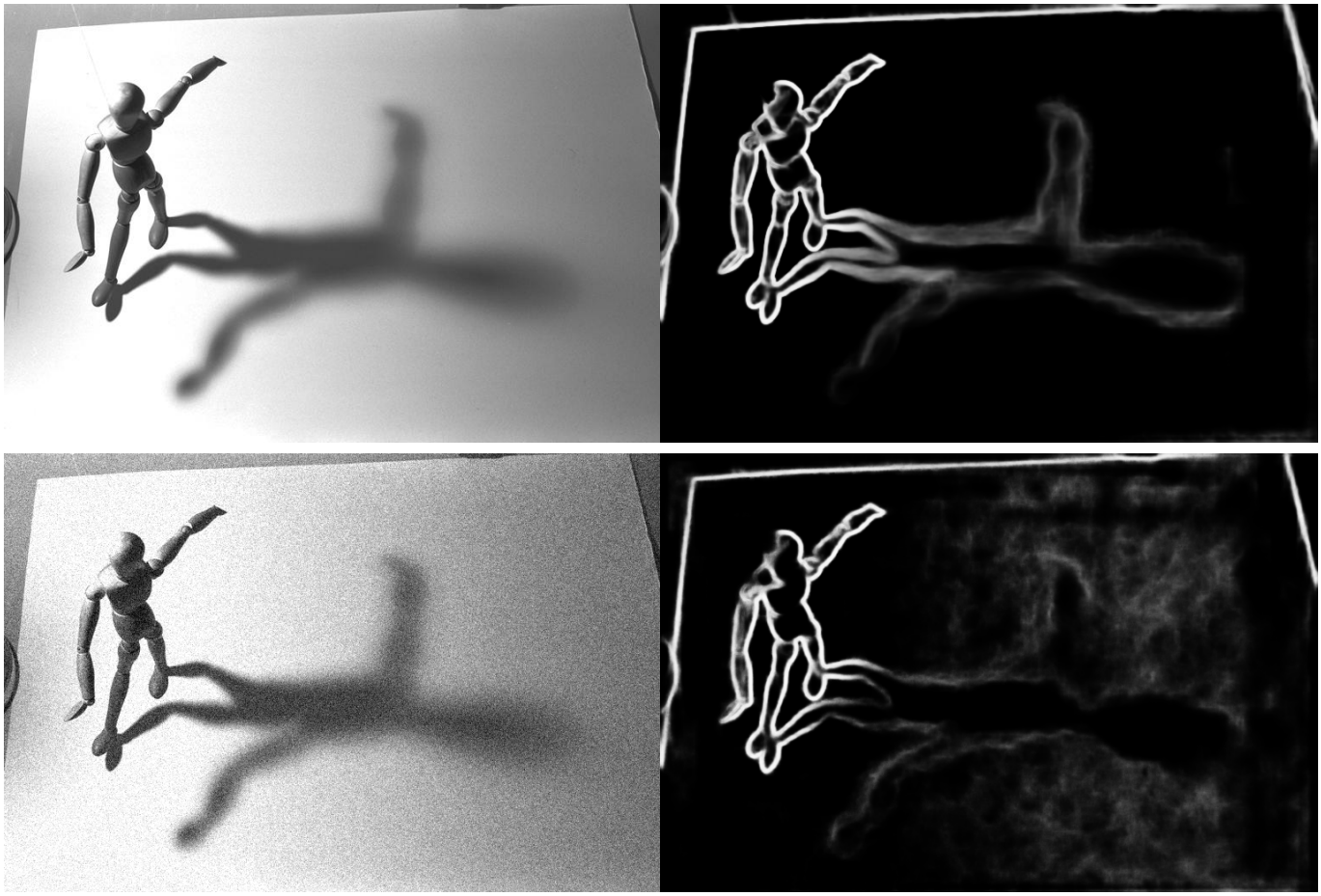


Figure 5: Input and result of HED on an image extracted from [2] of size 768×512. Note the effects of scale change and noise in the results. First row: original image and result of HED. Second row: Gaussian noise of $\sigma = 20$ was added to the original image.

provided by each human which consists in binary edge maps indicating the presence or not of edges. HED is thus not approximating the individual human annotators but the consensus among several human annotators. In many applications this is a very useful information. When a binary edge map is required, it can be obtained, for example, by applying a threshold on the output of HED.

HED does not differentiate edges from ridges and the result is not completely sharp (edge results are usually about 5 pixel wide). The presence of moderate noise may degrade the detection. Also, high-frequency patterns fail to produce detections. Finally, the response of HED is not fully isotropic, as would be desired.

HED shares a common aspect with all supervised learning method: the results are strongly dependent on the training dataset [8]. Indeed, there is a high level of subjectivity in the annotation of contours by human subjects. At what point do certain strokes cease to be edges and become texture? Only object contours should be annotated or edges observed on objects must be annotated too? For instance, the stripes of a zebra should be annotated or not? To which degree of detail should objects be classified as different? Should the clothes of a person be considered as separate objects from the person? One can argue that these problems may be essentially solved using a sufficient large annotated dataset. The BSDS300 dataset (or equivalently, the *train* and *val* sets of BSDS500) used to train HED, composed of just 300 images of size 481×321, is far from being large enough to capture the full complexity of the semantic analysis performed by the human annotators. HED was probably aided by initializing the network with the VGG16 parameters, which were trained on 1.3 million images, each with 1000 classes annotations [7]. Still, HED seems to have learned a mix

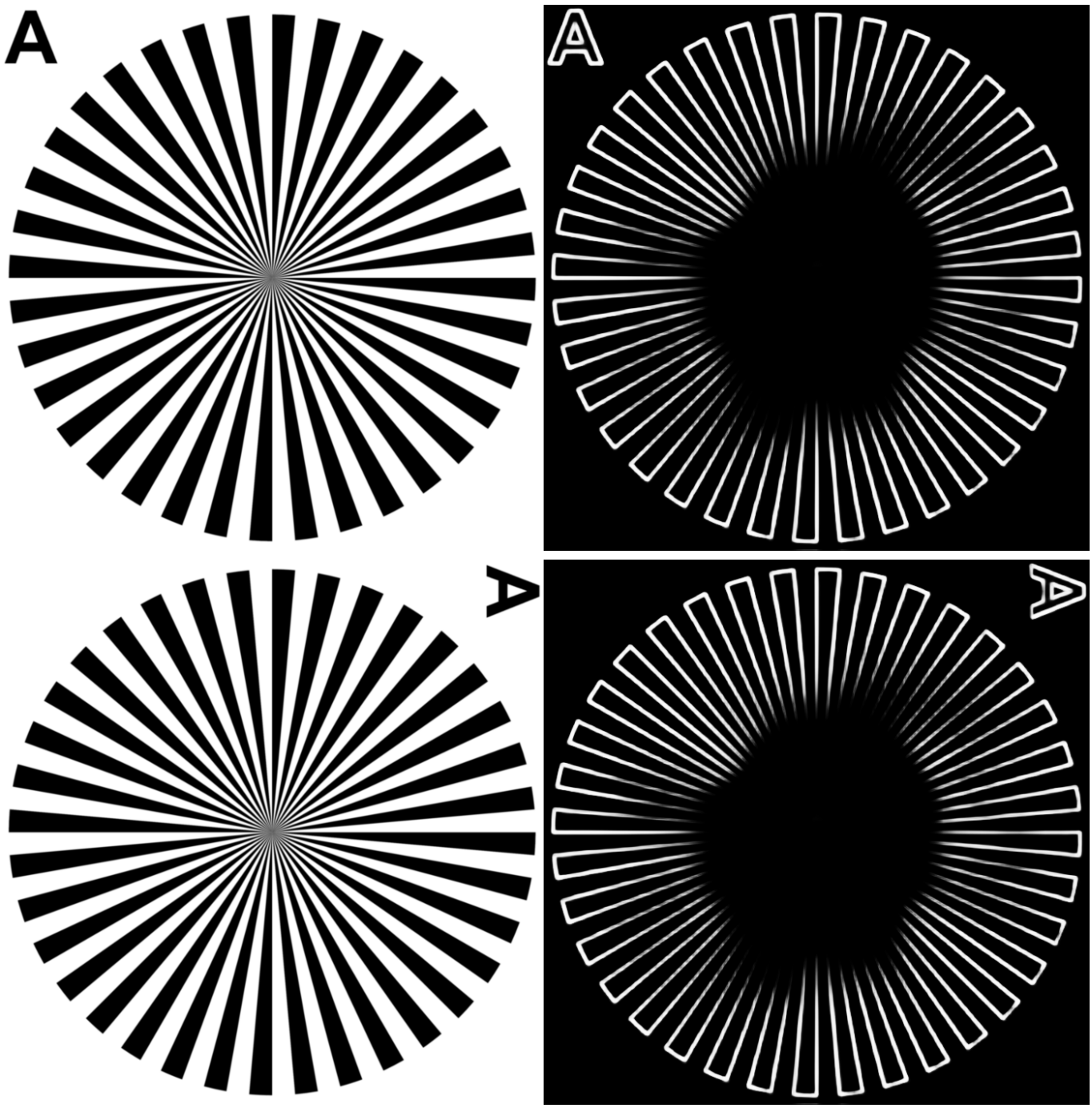


Figure 6: Experiments with a synthetic Siemens Star (700×700 pixels). First row: Input image and the result of HED. Second row: the same image as before turned 90 degrees (notice the orientation of the letter A); the result of HED shows the same anisotropic pattern as before, showing that it is due to HED and not to artifacts of the input image.

of semantic and geometrical analysis, probably dominated by the latter.

Taking into account all these considerations, the results of HED are quite impressive. All in all, HED is an excellent edge detector.

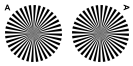
Image Credits

 Image number 35070 from BSDS500 database [1].

 Generated by the authors from data from BSDS500 database [1].

 From HED training set [9].

 Extracted from [2] without and with added noise.

 Synthetic images generated by the authors.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, *Contour detection and hierarchical image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33 (2010), pp. 898–916. https://doi.org/10.1109/TPAMI.2010.161.

[2] J.H. Elder and S.W. Zucker, *Local scale control for edge detection and blur estimation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (1998), pp. 699–716. https://doi.org/10.1109/34.689301.

[3] X.-Y. Gong, H. Su, D. Xu, Z. Zhang, F. Shen, and H-B. Yang, *An overview of contour detection approaches*, International Journal of Automation and Computing, 15 (2018), pp. 1–17. https://doi.org/10.1007/s11633-018-1117-z.

[4] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in IEEE International Conference on Computer Vision (ICCV), vol. 2, July 2001, pp. 416–423. https://doi.org/10.1109/ICCV.2001.937655.

[5] Simon N., *A reimplementation of HED using PyTorch*, 2018. https://github.com/sniklaus/pytorch-hed.

[6] G. Papari and N. Petkov, *Edge and line oriented contour detection: State of the art*, Image and Vision Computing, 29 (2011), pp. 79–103. https://doi.org/10.1016/j.imavis.2010.08.009.

[7] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, in International Conference on Learning Representations (ICLR), 2015. https://doi.org/10.48550/arXiv.1409.1556.

[8] X. Soria Poma, A. Sappa, P. Humanante, and A. Arbarinia, *Dense extreme Inception network for edge detection*, arXiv preprint arXiv:2112.02250, (2021). https://doi.org/10.48550/arXiv.2112.02250.

[9] S. Xie and Z. Tu, *Holistically-nested edge detection*, in IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1395–1403. https://doi.org/10.1109/ICCV.2015.164.