



Bi-Directional Pyramid Network for Edge Detection

Kai Li ¹, Yingjie Tian ^{2,3,*} , Bo Wang ^{4,*} , Zhiquan Qi ^{2,3} and Qi Wang ⁵

¹ School of Mathematics Sciences, University of Chinese Academy of Sciences, No.19, Yuquan Road, Shijingshan District, Beijing 100049, China; likai14@mailsucas.ac.cn

² Research Center on Fictitious Economy and Data Science, Chinese Academy of Sciences, No.80, Zhongguancun East Road, Beijing 100190, China; qizhiquan@foxmail.com

³ Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, No.80, Zhongguancun East Road, Beijing 100190, China

⁴ School of Information Technology and Management, University of International Business and Economics, No.10, Huixin Dongjie, Chaoyang District, Beijing 100029, China

⁵ China Mobile Research Institute, No.32, Xuanwumen West Street, Xicheng District, Beijing 100053, China; wangqi173@mailsucas.ac.cn

* Correspondence: tyj@ucas.ac.cn (Y.T.); wangbo@uibe.edu.cn (B.W.)

Abstract: Multi-scale representation plays a critical role in the field of edge detection. However, most of the existing research focuses on one of two aspects: fast training and accurate testing. In this paper, we propose a novel multi-scale method to resolve the balance between them. Specifically, according to multi-stream structures and the image pyramid principle, we construct a down-sampling pyramid network and a lightweight up-sampling pyramid network to enrich the multi-scale representation from the encoder and decoder, respectively. Next, these two pyramid networks and a backbone network constitute our overall architecture, a bi-directional pyramid network (BDP-Net). Extensive experiments show that compared with the state-of-the-art model, our method could improve the training speed by about one time while retaining a similar test accuracy. Especially, under the single-scale test, our approach also reaches human perception (F_1 score of 0.803) on the BSDS500 database.

Keywords: edge detection; encoder–decoder; multi-scale representation



Citation: Li, K.; Tian, Y.; Wang, B.; Qi, Z.; Wang, Q. Bi-Directional Pyramid Network for Edge Detection. *Electronics* **2021**, *10*, 329. <https://doi.org/10.3390/electronics10030329>

Academic Editor: Gwanggil Jeon
Received: 28 November 2020
Accepted: 7 January 2021
Published: 1 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Edge detection aims to extract perceptually salient edges and object boundaries from natural images. As a fundamental problem of image processing, edge detection has a significant impact on high-level feature extraction, feature description, and image understanding. Thus, it is closely related to many computer vision problems, including object recognition [1], image segmentation [2,3], and medical imaging [4,5].

Multi-scale representation learning is a long-standing topic in the field of computer vision, including edge detection. Edge detection is initially considered to be a low-level task until [2] applies local cues of multiple sizes to attain state-of-the-art at that time. In recent years, with the success of deep convolutional neural networks in computer vision, including object detection [6,7] and semantic segmentation [8–10], some multi-scale learning methods to fuse multiple kinds of hierarchical features for edge detection have been developed. For example, HED [11] utilizes deeply supervised nets [12] to supervise outputs of different network stages. Based on this, RCF [13] makes further efforts to enrich the features of different levels in the same network stage. Nevertheless, [11,13] do not fully take advantage of the contextual information of objects in a natural image, since constraints on the neighboring pixel labels are not directly enforced in their encoder architectures. On the basis of RCF, BDCN [14] attempts to resolve this issue by a pseudo bi-directional cascade structure (BDC) and scale enhancement module (SEM). As SEM can capture rich spatial contexts through multi-stream learning in the same network stage, it plays a major role in making BDCN state-of-the-art. Specifically, this module is implemented by multiple

dilated convolutions with various sampling rates. However, SEM needs to be utilized in every network stage, so it introduces some additional parameters and significantly increases training time.

To specify the performance of different methods, namely test accuracy and training speed, we show some relevant results in Table 1. Note that RCF [13] has an advantage over its previous approaches (including HED [11]) in terms of model performance. Therefore, here we do not show the performance of its previous methods. From Table 1, one may find that RCF and BDCN focus on one of two aspects: fast training and accurate testing, respectively. In order to take both factors into account, we attempt to consider a novel multi-scale learning method to deal with edge detection of natural images. To describe our architecture clearly, we start from the encoder and decoder, respectively.

Table 1. Single-scale test performance of different networks with 15 K training iterations on BSDS500. Here, ‘–’ represents ‘not used’. The metrics ODS and OIS refer to F₁ score based on optimal dataset scale and optimal image scale, respectively. Note that ‘#para’ refers to the model parameters and ‘+’ means the additional parameters on the basis of RCF [13]. ‘GFLOPS’ indicates giga floating-point operations per second. ‘TT’ and ‘FPS’ represent the training time of the model and the average frames per second during training, respectively.

Methods	ODS	OIS	# Para	GFLOPS	TT	FPS
Human	0.803	0.803	–	–	–	–
RCF [13]	0.792	0.809	14.8 M	81.77	7 h	5.95
BDCN_SEM [14]	0.801	0.817	+0.1 K	81.78	8 h 28 min	4.92
BDCN_BDC [14]	0.803	0.819	+1.5 M	114.3	12 h 27 min	3.35
BDCN [14]	0.804	0.820	+1.5 M	114.31	12 h 35 min	3.31
Ours	0.803	0.822	+3.9 K	81.81	7 h 42 min	5.41

First, we reconsider how to strengthen the encoder. Besides the multi-stream learning approach in the same network stage, like SEM, the one across network stages often appears in computer vision, such as [9,15]. However, this usually also consumes magnitude computations because of frequent up-sampling in the encoder. For a given image, the computational complexity of up-sampling is significantly higher than that of down-sampling. Intuitively, a feasible alternative is to apply down-sampling to multi-stream learning across network stages. Meanwhile, the amount of down-sampling had better not exceed a certain threshold, such as the total number of network stages. Based on the above ideas, we propose a simple and naive down-sampling pyramid network (DSP). Specifically, based on the image pyramid principle, according to the number difference between the current network stage and the first stage, we transform the original input image into a lower resolution one, and then fuse the down-sampling image and the corresponding features from the backbone to heighten the previous encoder further.

Moreover, we consider how to enhance the decoder architecture. Instead of simply utilizing channel concatenation like RCF [13], inspired by U-Net [16], we propose an up-sampling pyramid network (USP) to fuse information from different network stages. In particular, we restore the resolutions of feature maps gradually and then hierarchical features in our USP are separately fused with the corresponding ones in the encoder. To make our USP lightweight, we compress the feature maps in the encoder by a series of 1×1 convolutions. Finally, the triple mechanisms (i.e., our DSP, lightweight USP, and a trimmed VGG16 as the backbone) make up our bi-directional pyramid network (*BDP-Net*).

To sum up, our contributions are three-fold:

- Firstly, a down-sampling pyramid network is proposed to enrich the multi-scale representation of the previous encoder. To our knowledge, it is the first time that a down-sampling pyramid network has been used to enrich multi-scale features in the field of edge detection.

- Secondly, a *lightweight* up-sampling pyramid network is proposed to enhance the multi-scale representation of the previous decoder. Combining these two pyramid networks with a trimmed VGG16 makes up our bi-directional pyramid network (*BDP-Net*).
- Last but not least, while retaining the test accuracy with BDCN [14], the state-of-the-art model in the field of edge detection, the proposed BDP-Net is experimentally demonstrated to improve the training speed by about *one time*.

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 introduces our method and then Section 4 experimentally demonstrates its effectiveness and compares it with some state-of-the-art approaches. Finally, Section 5 provides a summary of the paper.

2. Related Work

Since our work is mainly related to multi-scale representation learning as well as edge detection, we give a brief review of these in the following.

2.1. Edge Detection

Edge detection is a long-standing topic dating back to work [17] in 1959. Since then, extensive literature has emerged about edge detection. Generally, one may divide these approaches into three categories: edge differential operators, traditional machine learning methods, and systems based on deep learning.

Edge differential operators are early pioneering methods (e.g., Robinson [18] and Sobel [19]), which mainly depend on the information of color and intensity. Specifically, they refer to the first or second derivative information. By introducing the idea of non-maximum suppression to edge detection, Canny [20] is widely used across various tasks and turns into a representative of such methods. However, since these methods rely on the manual design of edge detection operators instead of learnable patterns, they usually have poor accuracy on test data.

The traditional machine learning approaches, such as [2,21–25], generally apply data-driven methods to supervised learning by virtue of manual features. For instance, Martin et al. [21] formulate some low features about color, brightness, and texture in natural images, and then combine the information and their ground truth to train a classifier. Dollár et al. [24] make full use of predictions of local edge masks based on structured learning. Sam et al. [25] propose an efficient model, oriented edge forest, by combining a random forest classifier and a clustering method based on local edge orientations. Nevertheless, the edge information extracted from these methods is still relatively limited, even if designing sophisticated feature engineering.

Due to the emphasis on learning features automatically, methods based on deep learning, especially convolution neural networks (CNNs), have become the current mainstream in the field of edge detection. The early approaches based on deep learning, such as DeepContour [26], DeepEdge [27] and N⁴-Fields [28], mainly utilize patch-to-patch or patch-to-pixel strategies, which generally weaken prediction effectiveness. HED [11] first implements image-to-image prediction by leveraging fully convolutional neural networks. RCF [13] makes further efforts to enrich the hierarchical features by numerous 1×1 convolutions. BDCN [14] relies on a bi-directional pseudo-cascade structure and scale enhancement module to push forward state-of-the-art. In addition, LPCB [29] proposes a bottom-up/top-down architecture with a combination of cross-entropy and the Dice loss, to achieve good prediction for edge detection. However, as mentioned in Section 1, these methods are either insufficient in context information or inefficient in training.

2.2. Existing Multi-Scale Learning

Extracting and fusing multi-scale features is an enduring topic in computer vision, including edge detection. According to different locations used in a deep learning architec-

ture, multi-scale methods are divided roughly into two groups: those outside the network and those within the network.

The representative methods outside the network apply multi-scale inputs to a shared network or multiple networks and then fuse the multi-scale outputs. For example, Arbeláez et al. [30] begin with a multi-resolution image pyramid to perform multi-scale combinatorial grouping. Clément et al. [31] transform the original images through a Laplacian pyramid into multi-scale images that are fed to different networks. In addition to training, multi-scale inputs could be used for testing. For instance, Liu et al. [13] resize an original image to three resolutions, and then these are separately fed to the same network to obtain three outputs. Finally, these outputs are first restored to the original size and then integrated to achieve high-quality edge maps through a simple average. Although the multi-scale input method can improve the test accuracy, it also increases model training or testing time.

From the perspective of network stages, the multi-scale methods used within the network (see Figure 1) could be further divided into two groups, i.e., those used in the same stage and those applied in different stages.

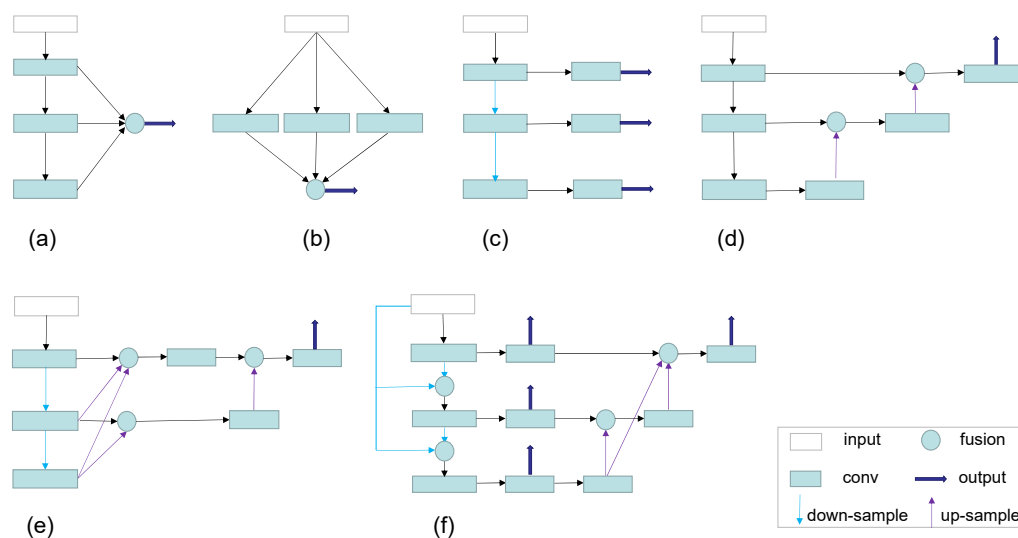


Figure 1. Outlines of multi-scale methods within the network: (a) multi-level learning; (b) modular structure; (c) multi-scale formed by different network stages; (d) encoder–decoder combined with skip-layers; (e) multi-scale across network stages; (f) our proposed bi-directional pyramid network (BDP-Net). Here, ‘conv’ refers to a convolution layer that usually embodies an activation operation. For simplicity, we show only a single-stage network for (a,b). Note that a dark blue (up or right) arrow indicates an output of the network.

In views of different forms, we divide multi-scale approaches used in the same network stage into two categories:

(a) Multi-level learning. In relation to a specific network, convolutions of different depths at the same stage have various receptive fields and then learn features of different levels. A typical example is RCF [13]. However, as mentioned in Section 1, RCF underutilizes the contextual information of objects in a natural image.

(b) Modular structure. Due to strong scalability, a large number of relevant methods have emerged in recent years, such as the Inception Series [32–34], Pyramid Pooling Module [35], ASPP [36], and SEM [14]. In addition to such parallel structures, some series structures like dense blocks [37] have been introduced. Generally, the modular structure could obtain good test results at the cost of increasing the model parameters and massive computation during training.

In relation to multi-scale methods applied in the different network stages, we divide these into the following categories:

(c) Multi-scale formed by different network stages. Generally, a deeper network stage often corresponds to a larger receptive field. For instance, Xie et al. [11] leverage the side-output layers to learn nested multi-scale features for edge detection. Lin et al. [6] introduce multi-scale outputs from different stages in the up-sampling feature pyramid. Due to a lack of sufficient context information in their encoder, their test accuracy still needs to improve.

(d) Encoder–decoder combined with skip-layers. For instance, U-Net [16] makes full use of the same level information from the encoder and decoder architecture. However, its decoder and encoder are almost the same in structure, which results in redundant parameters and calculations.

(e) Multi-scale across network stages. Hou et al. [15] and Zhang et al. [9] fuse information derived from the current stage and all the deeper stages and then provide a performance improvement. However, due to frequent up-sampling, their methods usually consume magnitude computations.

(f) Our architecture (BDP-Net). A down-sampling and lightweight up-sampling pyramid network are introduced. Contrary to [9,15], we up-sample two times at almost every network stage, which is a result of the down-sampling of the original input. Further, the proposed architecture could cut down many calculations and speed up the training process. Next, we introduce our method in detail.

3. Methodology

In this section, we introduce our architecture based on VGG16 [38]. In comparison with the raw VGG16, we not only remove all the fully connected layers and the 5th pooling operation, but also introduce modifications for the encoder and decoder as follows.

3.1. The Proposed Down-Sampling Pyramid Network

To make full use of contextual information in natural images, inspired by the multi-scale learning approaches [9,15] across network stages, we introduce the down-sampling pyramid network (DSP) by the image pyramid principle. Specifically, we propose some bypass networks to down-sample the original image and then these parallel branches constitute our DSP, as shown on the left of Figure 2. Note that one may conduct the above DSP by 1×1 convolutions with different strides, or combining 1×1 convolutions with stride 1 and bilinear (or bicubic) interpolation. The down-sampling stride is obtained according to the number difference between the current stage and the first stage. (Note that we adopt 1 as the stride of the 4th pooling like Liu et al. [13]. Thus, the stride of the last 1×1 convolution in the down-sampling pyramid network is 8, not 16. In addition, the last information fusion is before the 4th pooling operation.)

Next, the current down-sampling image is fused with the output from the previous stage in the backbone. Finally, the fusion information is fed to the next stage in our encoder. For example, the inputs of the third stage in our encoder contain the output of the second stage and the feature map obtained by 2^{3-1} down-sampling about the original image.

Further, the modified encoder is composed of this DSP and the previous encoder. Compared with the state-of-the-art model BDCN [14], our proposed DSP introduces no additional hyper-parameters. (Specifically, BDCN [14] needs to adjust two kinds of hyper-parameters, namely the number and the sampling rate of dilated convolutions.)

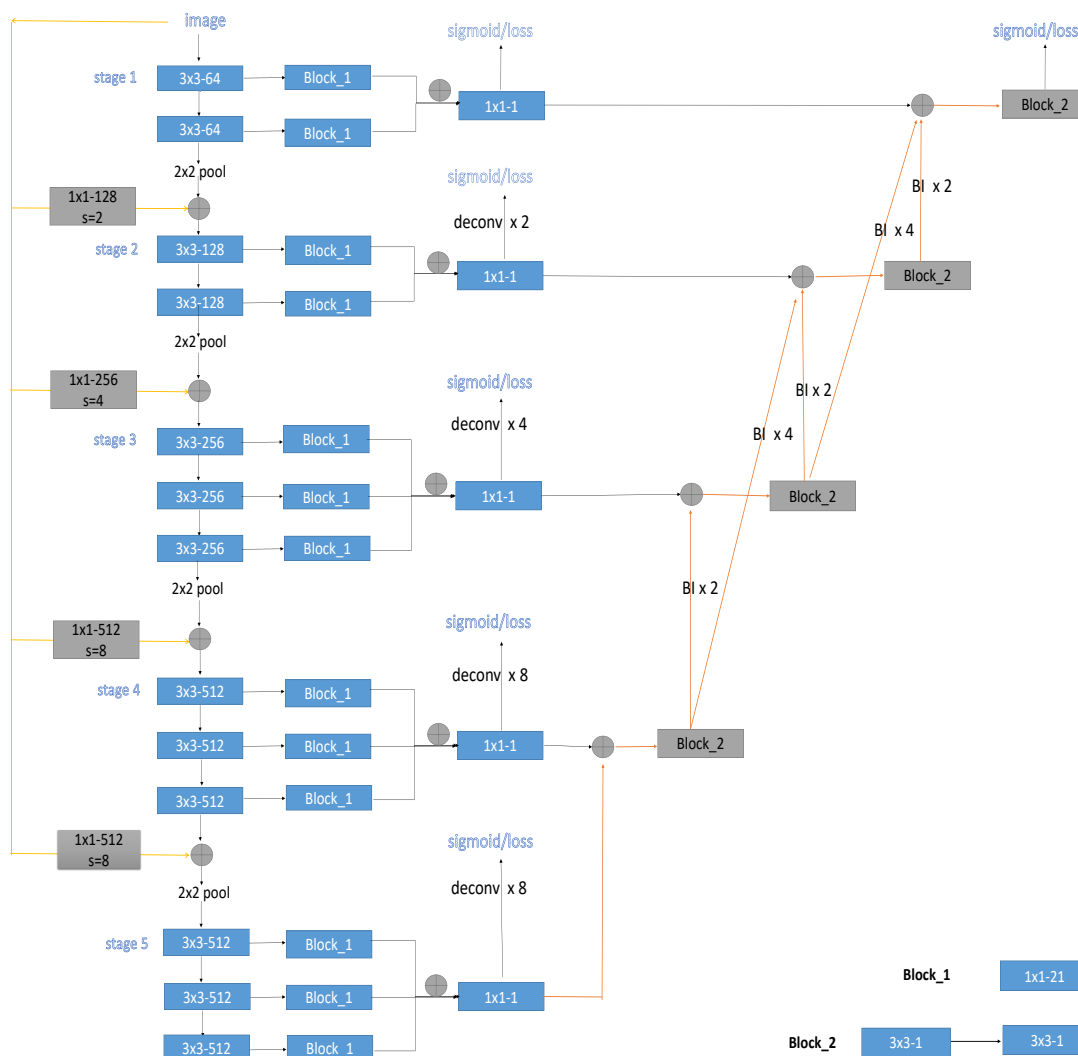


Figure 2. The framework of our BDP-Net. The left and right paths in orange correspond to our down-sampling pyramid network (DSP) and up-sampling pyramid network (USP), respectively. Here, ‘ $3 \times 3 - 64$ ’ represents a convolution with kernel 3 and channel 64, and ‘ 2×2 pool’ means a max-pooling with stride 2 in height and width. ‘ $s = 2$ ’ indicates that the down-sampling stride is 2. ‘ \oplus ’ means information fusion. ‘deconv $\times 2$ ’ and ‘BI $\times 2$ ’ refer to twice up-sampling using deconvolution and bilinear interpolation, respectively. The rest are similar. In addition, ‘sigmoid/loss’ indicates using the sigmoid layer to obtain a prediction and then calculating the corresponding loss.

3.2. The Proposed Lightweight Up-Sampling Pyramid Network

In the field of edge detection, RCF [13] makes use of abundant 1×1 convolutions to compress channel information and obtains rich hierarchical features by five side-outputs and a fusion output. In terms of the fusion output, RCF directly restores the resolutions of feature maps from different stages to the original input size and then applies one channel concatenation and one convolution operation to obtain the output. Evidently, such a fusion output in RCF is too simple to get rich information. BDCN [14] also has this problem, because it utilizes a fusion output similar to RCF.

Motivated by U-Net [16], we propose the lightweight up-sampling pyramid network (USP) to enrich multi-scale features from the decoder. Specifically, we take double and possible quadruple up-sampling for the feature map in the current phase (referring to stages 2, 3, and 4) and then fuse the information from the corresponding stage in our encoder, as shown on the right of Figure 2. Note that the fusion method involves adding between feature maps instead of concatenation. Hence, the channels from our decoder are the same (i.e., 1), as shown in Block_2 from Figure 2, which leads to a lightweight decoder.

Finally, our DSP, lightweight USP, and the trimmed VGG16 as the backbone, make up a novel encoder–decoder architecture: the bi-directional pyramid network (BDP-Net). One can see Section 4.3 for details of the model parameters and training speed of the proposed network. The adopted loss function originates from [14].

4. Experiments

In this section, we start with three datasets, state details of the implementation in our experiments, and then discuss the effectiveness of our network and make comparisons with some state-of-the-art models.

4.1. Datasets

We evaluate the proposed method on three public datasets: BSDS500 [2], NYUDv2 [39], and Multicue [40].

BSDS500 consists of 200, 100, and 200 images for training, validation, and testing, respectively. The ground-truth is obtained by average, as multiple annotators label every image. Its training and validation sets are used to finetune, and its test set is applied for evaluation. In relation to data augmentation, we utilize the same strategies as [11,13,14], which embrace three data transformations, namely scaling, rotating, and flipping. To expand the training set further, we also increase the flipped PASCAL VOC Context dataset.

NYUDv2 contains 381, 414, and 654 images for training, validation, and testing, respectively. Its ground truth provides binary annotations. We utilize the training and validation sets for finetuning, and the test set for evaluation. The data augmentation strategy is the same as [11,13,14]. Unlike BSDS500, it includes paired depth images in addition to RGB images, as it is initially applied for scene understanding. Following previous work [11], we apply this information by HHA features, namely horizontal disparity, height above ground, and angle with gravity.

Multicue includes 100 challenging natural scenes. There are ten frames taken from the left and right views in every scene. The last frame of the left view for each scene is annotated for two annotations, namely object boundaries and edges. Here we utilize 80 and 20 images for training and testing, respectively. Regarding data augmentation, we apply the same strategy as [13,14].

4.2. Implementation Details

On the three databases, the batch size was set to 10 for all the experiments. In the training phase, we applied full resolution images for BSDS500 and NYUDv2, and randomly cropped 500×500 patches for Multicue as its image resolution is high.

All experiments were conducted on a single GeForce GTX 1080Ti. Our architecture was implemented by using the publicly available PyTorch (<https://github.com/pytorch/pytorch>). In the experiments, the VGG16 [38] pre-trained on ImageNet [41] was applied to initialize the backbone.

SGD was utilized to train in this paper. The initial learning rate was 10^{-6} except 10^{-7} for edge detection on Multicue. The learning rate decayed by 10 times after every 10k iterations. Furthermore, momentum and weight decay were set to 0.9 and 2×10^{-4} , respectively. We trained 15 K iterations for BSDS500 and NYUDv2, 2 K and 1 K iterations for Multicue boundary and edge, respectively.

Standard non-maximum suppression is usually utilized to produce the thinned edge maps before evaluation. The maximum tolerance matching between edge predictions and ground-truth annotations was set to 0.0075 for BSDS500 and Multicue, and 0.011 for NYUDv2. In addition, the random seed was set to a fixed value of 7 to reduce the random error in our experiments.

Concerning evaluation metrics, we adopted the training speed (frame per second, abbreviated as FPS), the average precision (AP) on the full recall range (equivalently, the area under the precision-recall curve), as well as ODS and OIS, which correspond to

F_1 score based on the thresholds from the optimal dataset scale and optimal image scale, respectively. Note that the F_1 score is defined as

$$F_1 = \frac{2PR}{P + R}. \quad (1)$$

$$P = \frac{TP}{TP + FP}. \quad (2)$$

$$R = \frac{TP}{TP + FN}. \quad (3)$$

where TP, FP, and FN refer to true positive, false positive, and false negative, respectively. Here, P and R represent precision and recall, respectively. In addition, we utilize the evaluation metric introduced in [24], R50, which represents the recall at 50% precision.

We also discuss the computational complexity of different models. The computational complexity of a model usually includes spatial complexity and time complexity. Generally, one utilizes the number of floating-point operations per second (FLOPS) to represent the time complexity. In addition, one applies the parameters of the model to roughly obtain the spatial complexity.

4.3. The Effect of Network Architecture

In this section, we conduct experiments to discuss the effectiveness of the proposed network. For a fair comparison with RCF [13] and BDCN [14], we use the modified VGG16 as our backbone network.

4.3.1. Ablation Study on BSDS500

As our baseline, the modified VGG16 removes all the fully connected layers and the 5th pooling operation. Figure 3 reveals the single-scale test results on BSDS500. Compared with the baseline, the proposed down-sampling pyramid network (DSP) and light up-sampling pyramid network (USP) help to achieve higher ODS and OIS, respectively. Meanwhile, BDP-Net, as a combination of both mechanisms, could further improve the evaluation metrics.

In addition to the baseline model, we also compare our method with two state-of-the-art models, i.e., RCF [13] and BDCN [14], by reproducing their experiments. From Figure 3 and Table 2, we can see that:

(i) Our USP helps to achieve *better* generalization *earlier* than RCF. In particular, under the single-scale test, it achieves higher ODS and OIS (0.799 and 0.813) than RCF (0.792 and 0.809), and concurrently the training time is 4 h 55 min for 10 K iterations and 7 h for 15 K iterations, respectively.

(ii) Compared with BDCN, the proposed BDP-Net utilizes nearly *half* of the training time while retaining comparable metrics. Specifically, under 15K training iterations, BDP-Net needs 7 h 42 min to obtain ODS, OIS, and AP (0.803, 0.822, and 0.845) while BDCN requires 12 h 35 min to achieve the performance with the metrics (0.804, 0.820, and 0.724) under the single-scale test.

(iii) Note that our BDP-Net also *catches up with* human perception (0.803) under the single-scale test while using only the training set from BSDS500. In terms of the training speed, BDP-Net is nearly twice as fast as BDCN [14].

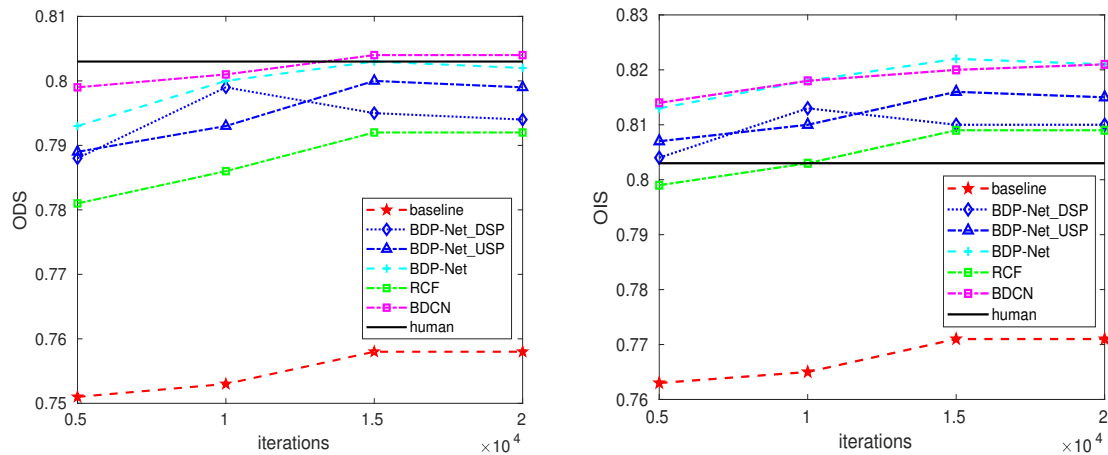


Figure 3. Evaluation results on BSDS500. Here, we utilize a single-scale test. The solid pink line represents the F₁ score (0.803) from human perception. BDP-Net_DSP and BDP-Net_USP refer to our BDP-Net without the down-sampling and up-sampling pyramid networks, respectively.

Table 2. Evaluation results on BSDS500. They are obtained with 15 K training iterations except that those of BDP-Net_DSP are trained with 10 K iterations. Note that ‘–’ represents ‘not used’; ‘# para’ refers to the model parameters, and ‘+’ means the added parameters on the basis of RCF [13]. ‘GFLOPS’ indicates giga floating-point operations per second. In addition, the superscript ‘+’ corresponds to the multi-scale test versions.

Methods	ODS	OIS	AP	R50	# Para	GFLOPS	TT	FPS
Human	0.803	0.803	–	–	–	–	–	–
Baseline	0.758	0.771	0.673	0.773	–89,685	80.49	5 h 25 min	7.69
BDP-Net_DSP	0.798	0.813	0.714	0.899	+74	81.78	4 h 55 min	5.65
BDP-Net_USP	0.800	0.816	0.772	0.889	+3840	81.8	7 h 9 min	5.83
BDP-Net	0.803	0.822	0.845	0.918	+3914	81.81	7 h 42 min	5.41
Baseline ⁺	0.788	0.799	0.766	0.812	–89,685	80.49	5 h 25 min	7.69
BDP-Net_DSP ⁺	0.807	0.823	0.795	0.902	+74	81.78	4 h 55 min	5.71
BDP-Net_USP ⁺	0.807	0.825	0.804	0.889	+3840	81.8	7 h 9 min	5.83
BDP-Net ⁺	0.808	0.828	0.847	0.913	+3914	81.81	7 h 42 min	5.41
RCF [13]	0.792	0.809	0.780	0.898	14.8 M	81.77	7 h	5.95
BDCN [14]	0.804	0.820	0.724	0.894	+1.5 M	114.31	12 h 35 min	3.31
RCF ⁺ [13]	0.801	0.820	0.803	0.895	14.8 M	81.77	7 h	5.95
BDCN ⁺ [14]	0.811	0.831	0.796	0.890	+1.5 M	114.31	12 h 35 min	3.31

Moreover, compared with the USP, our DSP could achieve a similar test accuracy and training speed. In the following, we discuss the computational complexity of different models. Since using the same group of networks (e.g., RCF, BDCN, and our BDP-Net) and the same configurations on three public datasets, we only compare the model parameters through Table 2 in this section. To compare the parameters of different models conveniently, we utilize RCF as the base instead of the modified VGG16. In Table 2, compared with RCF (14.8 M parameters and 81.77 GFLOPS), our down-sampling and lightweight up-sampling pyramid network add negligible parameters and floating-point operations. Their combination, BDP-Net, has a smaller increase in complexity (3.9 K parameters and 0.04 GFLOPS). Meanwhile, BDCN requires significantly more parameters and floating point operations (1.5 M parameters and 32.54 GFLOPS), thus its training efficiency is greatly reduced.

Finally, we show the test accuracy of more methods in Figure 4. Specifically, apart from HED [11], RCF [13], BDCN [14], and our BDP-Net, it includes (1) differential operator, Canny [20]; (2) traditional machine learning methods, including gPb-UCM [2], Pb [21], EGB [23], SE [24], OEF [25], and MShift [42]; (3) early deep learning methods based on image patches, such as DeepContour [26] and DeepEdge [27]. Compared with the

non-deep-learning approaches, the deep learning methods show obvious advantages. Compared with the deep learning systems based on image patches, recent approaches (e.g., HED, RCF, and BDCN) combine with multi-scale learning and image-to-image training to further improve the test accuracy. In comparison with BDCN, the state-of-the-art, our BDP-Net achieves a similar test accuracy on the BSDS500 dataset. However, when adding more training data (e.g., PASCAL Context dataset [43]), our approach is slightly inferior, which sheds light on further improvement. In addition, we also display the prediction results obtained by different methods, as shown in Figure 5.

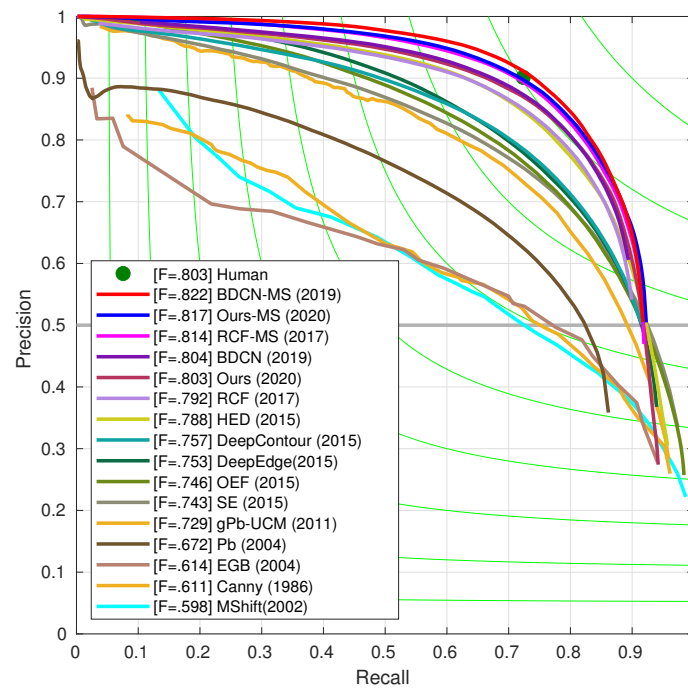


Figure 4. The precision–recall curves of various approaches using a single-scale test on BSDS500. We also display multi-scale versions of some approaches, i.e., BDCN-MS, Ours-MS, and RCF-MS, which are trained by utilizing an additional PASCAL Context dataset [43]. Note that the metrics precision and recall are obtained by using the optimal dataset scale during evaluation.

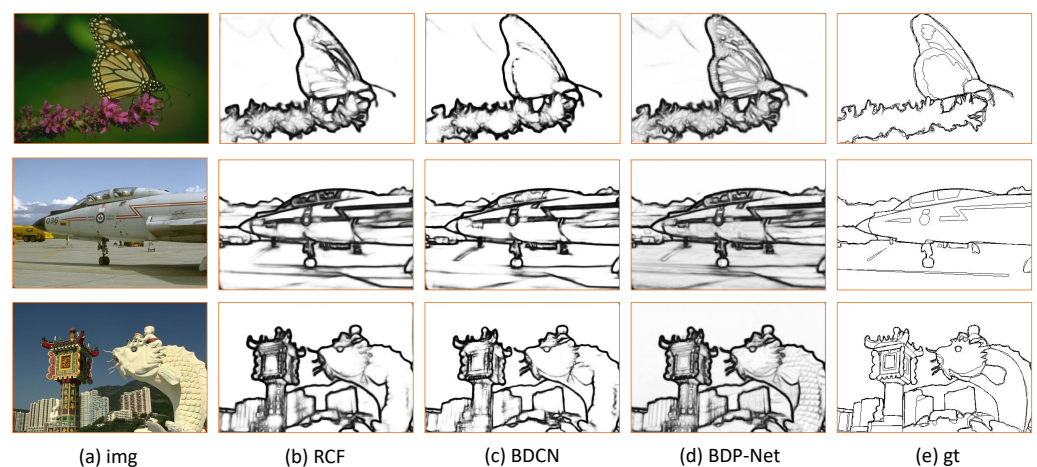


Figure 5. Visual comparison on BSDS500: (a) contains the raw images; (b–d) correspond to the results obtained by these networks; (e) corresponds to their ground-truth. For simplicity, we merely show the prediction results where the training set is the augmented BSDS500. Note that the prediction images are those before using non-maximum suppression.

4.3.2. Performance on NYUDv2 and Multicue

In terms of NYUDv2, we first performed experiments on RGB and HHA images separately, and then averaged their predictions to obtain the results of RGB-HHA. Table 3 shows the evaluation metrics of RCF [13], BDCN [14], and our BDP-Net on NYUDv2. In relation to HHA feature images, there is little difference for the metrics (ODS, OIS, AP, and R50) obtained by leveraging three different networks. For RGB images, our method has a slight advantage in terms of average precision. In addition, compared with RCF, our BDP-Net has an approximate floating point operation and training speed while BDCN adds a large number of floating-point operations and then its training speed is significantly reduced. Compared with RCF and BDCN for NYUDv2 RGB images, our method gains an advantage in balancing effectiveness and efficiency. Please see Figure 6 for the precision–recall curves of different methods. We also display some visual results in Figure 7.

Table 3. Comparison of different methods with 15 K training iterations on NYUDv2. Here, we utilize the multi-scale test to get the following results.

Methods	Data	ODS	OIS	AP	R50	GFLOPS	TT	FPS
RCF [13]	HHA	0.693	0.709	0.678	0.827	94.21	13 h 26 min	3.10
	RGB	0.736	0.755	0.728	0.896	94.21	13 h 26 min	3.10
	RGB-HHA	0.751	0.770	0.765	0.899	-	-	-
BDCN [14]	HHA	0.694	0.708	0.671	0.818	131.39	24 h 26 min	1.71
	RGB	0.752	0.768	0.739	0.874	131.39	24 h 26 min	1.71
	RGB-HHA	0.762	0.776	0.761	0.869	-	-	-
BDP-Net	HHA	0.694	0.708	0.676	0.812	94.25	13 h 43 min	3.04
	RGB	0.746	0.762	0.750	0.892	94.25	13 h 43 min	3.04
	RGB-HHA	0.759	0.776	0.772	0.895	-	-	-

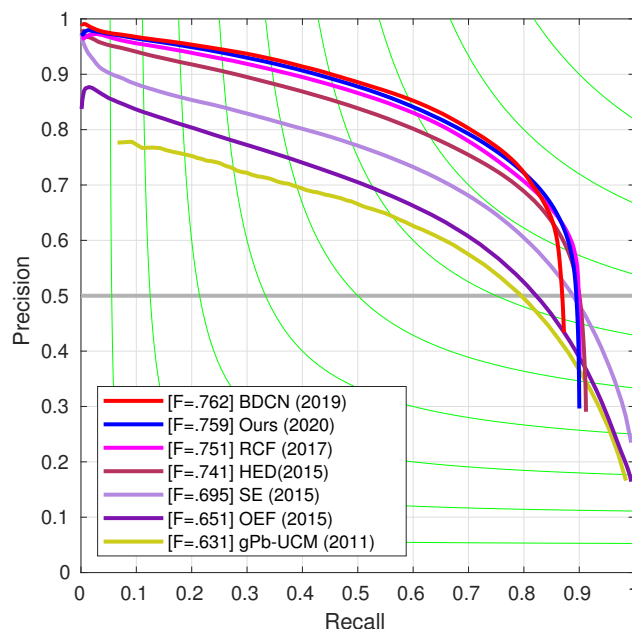


Figure 6. The precision–recall curves of various approaches on NYUDv2. Note that the metrics precision and recall are obtained by using the optimal dataset scale during evaluation.

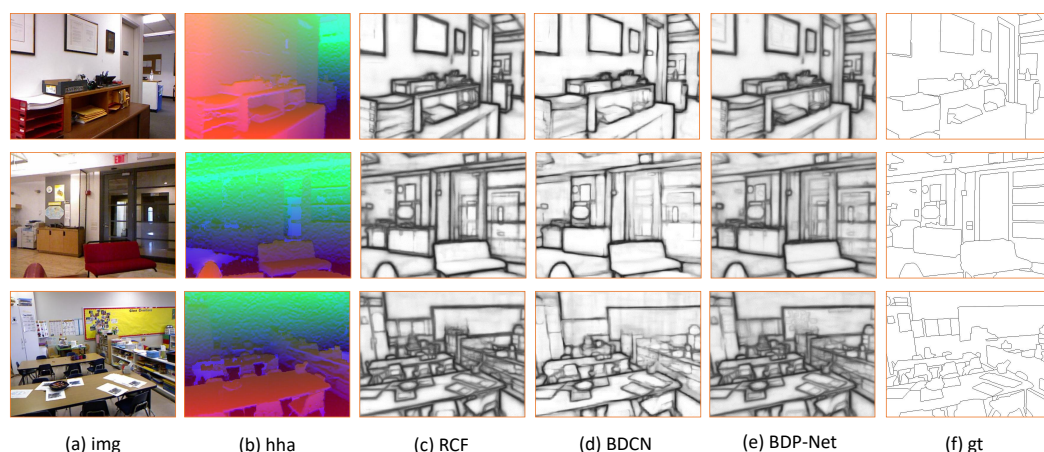


Figure 7. Visual comparison on NYUDv2: (a) contains the original images; (b) shows the corresponding HHA feature images; (c–e) are obtained by averaging the prediction results on RGB and HHA, respectively; (f) corresponds to their ground-truth.

For Multicue, we carried out two distinct visual tasks, namely object boundary and edge. Table 4 reveals some evaluation results of different architectures. Concerning object boundary, BDP-Net attains almost as good a performance with respect to the given metrics as BDCN [14] while retaining the similar training speed with RCF [13]. Meanwhile, compared with RCF and BDCN for edge detection, our BDP-Net achieves growth of about two points on the metrics (ODS, OIS, AP, and R50). In addition, compared with RCF, our approach has similar floating-point operations and slightly lower training speed. However, BDCN possesses a significant increase in floating-point operations. Moreover, it has more parameters, and thus its training efficiency is greatly reduced. All in all, it seems to have no advantage on such a small database. In addition, please see Figures 8 and 9 respectively for the precision–recall curves and the visual results of different methods for Multicue.

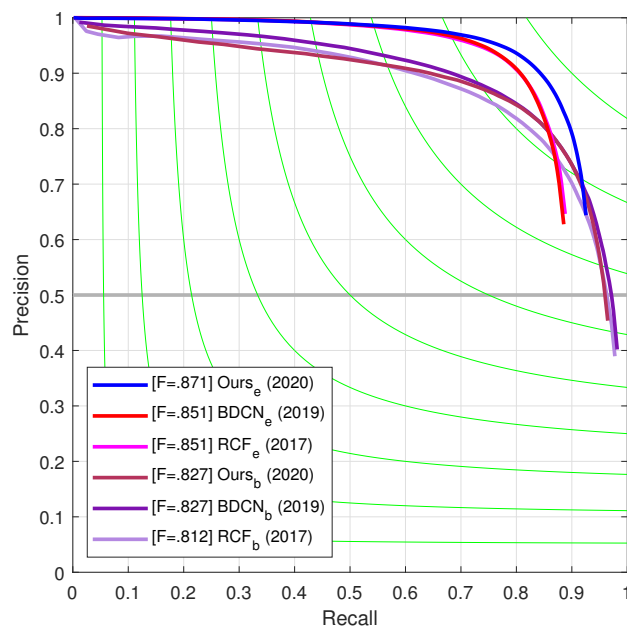


Figure 8. The precision–recall curves of various approaches on Multicue. The subscripts “e” and “b” represent the tasks of edge and boundary detection, respectively. Note that the metrics precision and recall are obtained by using the optimal dataset scale during evaluation.

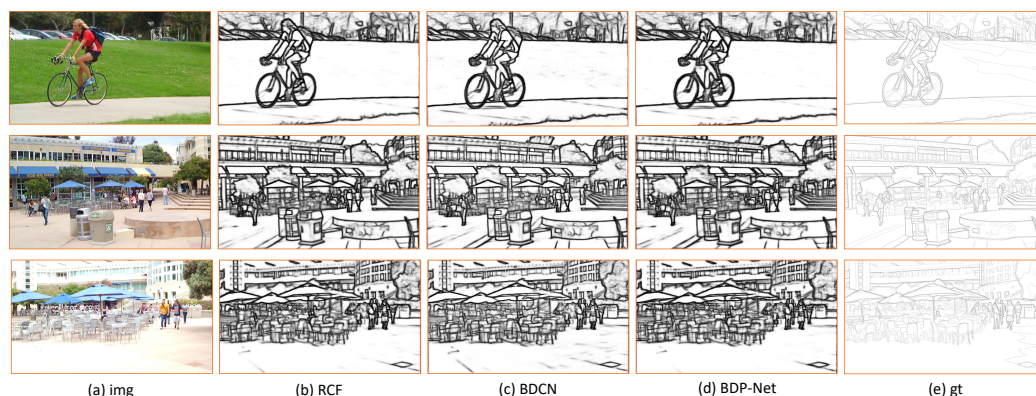


Figure 9. Visual comparison on the Multicue edge database: (a) contains the raw images; (b–d) correspond to the results obtained by these networks; (e) is their ground-truth for the Multicue edge task. To see these images and their ground-truth, you may need to zoom in at least 3 times, as they have high resolutions, and their annotations are rich and hierarchical.

Table 4. Comparison of different methods with 2 K and 1 K training iterations for Multicue boundary and edge, respectively. Here, the following results are derived by applying the multi-scale test.

Methods	ODS	OIS	AP	R50	GFLOPS	TT	FPS
Human_Boundary [40]	0.760	-	-	-	-	-	-
RCF_Boundary [13]	0.812	0.817	0.861	0.963	98.77	1 h 45 min	3.17
BDCN_Boundary [14]	0.827	0.835	0.875	0.971	137.78	3 h 15 min	1.71
BDP-Net_Boundary	0.827	0.834	0.842	0.974	98.81	1 h 48 min	3.09
Human_Edge [40]	0.750	-	-	-	-	-	-
RCF_Edge [13]	0.851	0.857	0.853	0.888	98.77	53 min	3.14
BDCN_Edge [14]	0.851	0.858	0.852	0.889	137.78	1 h 36 min	1.74
BDP-Net_Edge	0.871	0.877	0.892	0.926	98.81	1 h 6 min	2.53

5. Conclusions and Discussion

To sum up, in this paper, we developed a novel multi-scale learning method, BDP-Net, to deal with contextual information of objects in natural images. According to our extensive experiments, one can find that compared with the state-of-the-art model BDCN [14], our BDP-Net can cut down about *half* of the training time while retaining a similar test accuracy.

However, from Figure 4 and Table 3, we find that compared with BDCN, our BDP-Net still has some room for improvement in the test accuracy when enough training data are available. To our knowledge, common down-sampling methods, such as bilinear interpolation and 1×1 convolution with stride 2 or more, usually lose some spatial position information in natural images. Therefore, our down-sampling pyramid network also has a similar problem. Besides, to reduce the loss of spatial information, the three methods (namely our method, the previous RCF, and BDCN) apply down-sampling three instead of four or more times in the backbone network. Furthermore, their training speed still needs to improve. A possible solution could be to combine model compression with a transformer, such as in [44,45]. In addition, some modules from neural architecture search (NAS), such as Lite R-ASPP [46], potentially provide novel ideas for edge detection. However, these are out of the scope of this work. For more details of NAS modules, please refer to the relevant literature [47,48].

Author Contributions: Conceptualization, K.L.; methodology, K.L.; software, K.L.; validation, K.L.; formal analysis, K.L., B.W.; investigation, K.L., Q.W.; resources, K.L.; data curation, K.L.; writing—original draft preparation, K.L.; writing—review and editing, K.L., B.W., Q.W.; visualization, K.L.; supervision, Y.T., Z.Q.; project administration, Y.T., Z.Q.; funding acquisition, Y.T., B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by grants from the National Natural Science Foundation of China (Nos. 61702099, 12071458, and 71731009) and the Fundamental Research Funds for the Central Universities in UIBE (No. CXTD10-05).

Acknowledgments: The authors would like to thank all the editors and anonymous reviewers for their careful reading and insightful remarks.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ramadevi, Y.; Sridevi, T.; Poornima, B.; Kalyani, B. Segmentation and Object Recognition Using Edge Detection Techniques. *Int. J. Comput. Sci. Inf. Technol.* **2010**, *2*, 153–161.
2. Arbeláez, P.; Maire, M.; Fowlkes, C.; Malik, J. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 898–916.
3. Muthukrishnan, R.; Radha, M. Edge Detection Techniques for Image Segmentation. *Int. J. Comput. Sci. Inf. Technol.* **2011**, *3*, 259. [[CrossRef](#)]
4. Gudmundsson, M.; El-Kwae, E.A.; Kabuka, M.R. Edge Detection in Medical Images Using A Genetic Algorithm. *IEEE Trans. Med. Imaging* **1998**, *17*, 469–474. [[CrossRef](#)] [[PubMed](#)]
5. Nikolic, M.; Tuba, E.; Tuba, M. Edge Detection in Medical Ultrasound Images Using Adjusted Canny Edge Detection Algorithm. In Proceedings of the IEEE 24th Telecommunications Forum, Belgrade, Serbia, 22–23 November 2016; pp. 1–4.
6. Lin, T.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
7. Guo, C.; Fan, B.; Zhang, Q.; Xiang, S.; Pan, C. AugFPN: Improving Multi-Scale Feature Learning for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 12595–12604.
8. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–12 June 2015; pp. 3431–3440.
9. Zhang, Z.; Zhang, X.; Peng, C.; Cheng, D.; Sun, J. ExFuse: Enhancing Feature Fusion for Semantic Segmentation. In Proceedings of the IEEE Conference on European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 273–288.
10. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 801–818.
11. Xie, S.; Tu, Z. Holistically-Nested Edge Detection. In Proceedings of the IEEE International Conference on Computer Vision, San Diego, CA, USA, 13–16 December 2015; pp. 1395–1403.
12. Lee, C.; Xie, S.; Gallagher, P.W.; Zhang, Z.; Tu, Z. Deeply-Supervised Nets. In Proceedings of the AISTATS, San Diego, CA, USA, 9–12 May 2015; pp. 562–570.
13. Liu, Y.; Cheng, M.M.; Hu, X.; Wang, K.; Bai, X. Richer Convolutional Features for Edge Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 3000–3009.
14. He, J.; Zhang, S.; Yang, M.; Shan, Y.; Huang, T. Bi-Directional Cascade Network for Perceptual Edge Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–21 June 2019; pp. 3828–3837.
15. Hou, Q.; Liu, J.; Cheng, M.; Borji, A.; Torr, P.H.S. Three Birds One Stone: A Unified Framework for Salient Object Segmentation, Edge Detection and Skeleton Extraction. In Proceedings of the IEEE Conference on European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 1–17.
16. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
17. Julesz, B. A Method of Coding Television Signals Based on Edge Detection. *Bell Syst. Tech. J.* **1959**, *38*, 1001–1020. [[CrossRef](#)]
18. Robinson, G.S. Color Edge Detection. *Proc. SPIE* **1975**, *87*, 126–133. [[CrossRef](#)]
19. Sobel, I. *Camera Models and Machine Perception*; Technical Report; Department of Electrical Engineering, Stanford University: Stanford, CA, USA, 1970.
20. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *6*, 679–698. [[CrossRef](#)]
21. Martin, D.; Fowlkes, C.C.; Malik, J. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 530–549. [[CrossRef](#)] [[PubMed](#)]
22. Lim, J.J.; Zitnick, C.L.; Dollar, P. Sketch Tokens: A Learned Mid-level Representation for Contour and Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 3158–3165.
23. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vis.* **2004**, *59*, 167–181. [[CrossRef](#)]
24. Dollár, P.; Zitnick, C.L. Fast Edge Detection Using Structured Forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1558–1570. [[CrossRef](#)] [[PubMed](#)]

25. Hallman, S.; Fowlkes, C.C. Oriented Edge Forests for Boundary Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–12 June 2015; pp. 1732–1740.
26. Zhang, Z. DeepContour: A Deep Convolutional Feature Learned by Positive-sharing Loss for Contour Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–12 June 2015; pp. 3982–3991.
27. Torresani, L. DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–12 June 2015; pp. 4380–4389.
28. Ganin, Y.; Lempitsky, V. N^4 -Fields: Neural Network Nearest Neighbor Fields for Image Transforms. In Proceedings of the Asian Conference on Computer Vision, Singapore, 1–5 November 2014; pp. 536–551.
29. Deng, R.; Shen, C.; Liu, S.; Wang, H.; Liu, X. Learning to Predict Crisp Boundaries. In Proceedings of the IEEE Conference on European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 570–586.
30. Arbelaez, P.; Ponttuset, J.; Barron, J.; Marques, F.; Malik, J. Multiscale Combinatorial Grouping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 328–335.
31. Farabet, C.; Couprie, C.; Najman, L.; Lecun, Y. Learning Hierarchical Features for Scene Labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1915–1929. [[CrossRef](#)] [[PubMed](#)]
32. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–12 June 2015; pp. 1–9.
33. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
34. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4278–4284.
35. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6230–6239.
36. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834. [[CrossRef](#)] [[PubMed](#)]
37. Huang, G.; Liu, Z.; Der Maaten, L.V.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
38. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
39. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor Segmentation and Support Inference from RGBD Images. In Proceedings of the 12th European Conference on Computer Vision, Florence, Italy, 7–13 October 2012; pp. 746–760.
40. Mély, D.A.; Kim, J.; McGill, M.; Guo, Y.; Serre, T. A Systematic Comparison between Visual Cues for Boundary Detection. *Vis. Res.* **2016**, *120*, 93–107. [[CrossRef](#)] [[PubMed](#)]
41. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, F.F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, USA, 20–26 June 2009; pp. 1037–1037.
42. Comaniciu, D.; Meer, P. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 603–619. [[CrossRef](#)]
43. Mottaghi, R.; Chen, X.; Liu, X.; Cho, N.G.; Yuille, A. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In Proceedings of the Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 891–898.
44. So, D.R.; Liang, C.; Le, Q.V. The Evolved Transformer. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 5877–5886.
45. Wu, Z.; Liu, Z.; Lin, J.; Lin, Y.; Han, S. Lite Transformer with Long-Short Range Attention. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
46. Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; et al. Searching for MobileNetV3. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324.
47. Thomas, E.; Jan, H.M.; Frank, H. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1–21.
48. Ren, P.; Xiao, Y.; Chang, X.; Huang, P.Y.; Li, Z.; Chen, X.; Wang, X. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *arXiv* **2020**, arXiv:2006.02903.