Bottom-up and Top-down: Bidirectional Additive Net for Edge Detection

Lianli Gao, **Zhilong Zhou**, **Heng Tao Shen*** and **Jingkuan Song***
Center for Future Media and School of Computer Science and Technology, University of Electronic Science and Technology of China
lianli.gao@uestc.edu.cn, {zhilong.zhou1996, jingkuan.song}@gmail.com, shenhengtao@hotmail.com

Abstract

Image edge detection is considered as a cornerstone task in computer vision. Due to the nature of hierarchical representations learned in CNN, it is intuitive to design side networks utilizing the richer convolutional features to improve the edge detection. However, there is no consensus way to integrate the hierarchical information. In this paper, we propose an effective and end-to-end framework, named Bidirectional Additive Net (BAN), for image edge detection. In the proposed framework, we focus on two main problems: 1) how to design a universal network for incorporating hierarchical information sufficiently; and 2) how to achieve effective information flow between different stages and gradually improve the edge map stage by stage. To tackle these problems, we design a bottom-up and top-down architecture, where a bottom-up branch can gradually remove detailed or sharp boundaries to enable accurate edge detection and a top-down branch offers a chance of error-correcting by revisiting the low-level features that contain rich textual and spatial information. Attended additive module (AAM) is designed to cumulatively refine edges by selecting pivotal features in each stage. Experimental results show that our proposed method can improve the edge detection performance to new records and achieve state-of-the-art results on two public benchmarks: BSDS500 and NYUDv2.

1 Introduction

Edge detection aims to assign a label either edge pixel or non-edge pixel to each pixel given an image, which is considered as a fundamental task in computer vision and plays an important role in other higher-level tasks such as semantic segmentation [Chen *et al.*, 2016] and salient detection [Wang *et al.*, 2019a]. The history of researching edge detection is extremely long. Early researchers only focus on local cues such as texture gradients, colors, or other hand-crafted visual features [Martin *et al.*, 2004]. But the lack of high-level semantic information restricts the abilities to capture accurate edges.

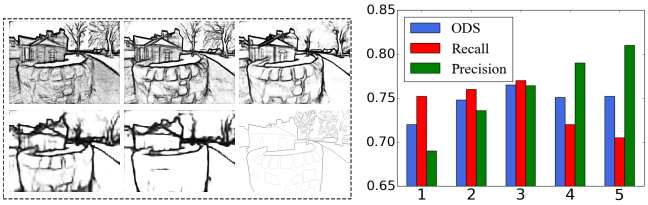


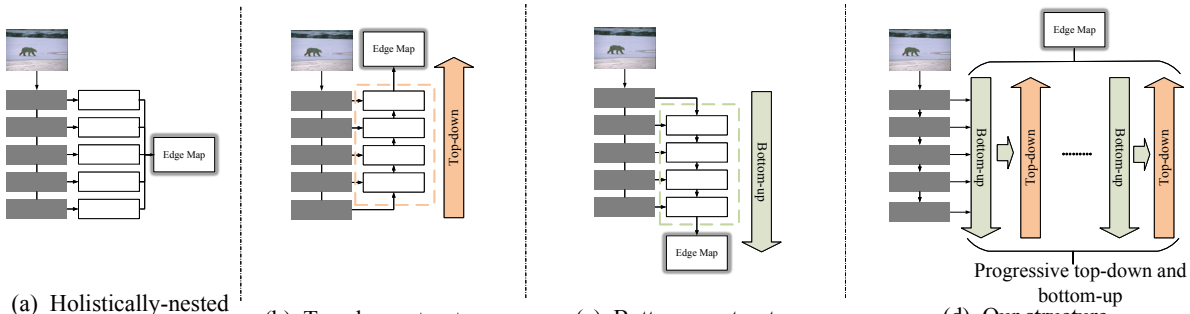
Figure 1: The quantitative and qualitative evaluations on different side-outputs.(stage1-5:bottom-up) (a): The edge maps predicted from inferior layers contain much details and noises, while those from upper layers have the ability to capture object contours. The precision and recall metrics in (b) have proved the conclusion, which motivates us to investigate multi-level representations.

More recently, deep learning has achieved impressive results for edge detection. A series of deep learning methods have been proposed, e.g., HED [Xie and Tu, 2017], RCF [Liu *et al.*, 2019], DeepEdge [Bertasius *et al.*, 2015], DeepContour [Shen *et al.*, 2015b], CEDN [Yang *et al.*, 2016], COB [Maninis *et al.*, 2018], CNet [Song *et al.*, 2018], LPCB [Deng *et al.*, 2018], AMH-Net [Xu *et al.*, 2017], CED [Wang *et al.*, 2019b] and BDCN [He *et al.*, 2019]. The previous deep learning methods can be grouped into three main categories generally: 1) holistically-nested architecture, 2) top-down architecture and 3) bottom-up architecture, as shown in Fig 2.

The holistically-nested architecture is employed in HED [Xie and Tu, 2017] and [Liu *et al.*, 2019], which extracts features from five convolutional stages of VGG-16 to obtain the side-outputs and the final result is generated by a fixed weight fusion strategy (Fig 2(a)). Generally, the low-level features are noisy but contain much spatial information, while the high-level features are coarse but carry much semantic information [Guo *et al.*, 2019; Gao *et al.*, 2019; Zhu *et al.*, 2016; Shen *et al.*, 2015a]. As shown in Fig 1, the visual edge maps from different stages and evaluation metrics verify this conclusion. Naturally, the hierarchical features are complementary for the edge detection. But a linear combination has limited ability to investigate the complex relationship between multi-level edge features since the side-outputs are isolated.

The top-down architecture in edge detection is inspired by the popular U-Net structure, in which rough high-level features carrying rich semantic information are progressively passed to lower layers (Fig 2(b)). In this way, the lost de-

*Corresponding Author



(a) Holistically-nested structure (b) Top-down structure (c) Bottom-up structure (d) Our structure

Figure 2: The comparison of proposed method with other architectures for edge detection task.

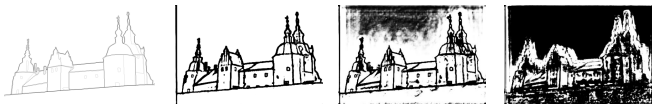


Figure 3: We visualize an example of the generated feature maps (the 1st, 7th and 14th channel). We can find the 1st slice has a more accurate prediction than others and the wrong information in other channels might affect the final estimation.

tailed boundaries will be retrieved and the coarse features maps will be refined gradually in this branch. The bottom-up architecture (Fig 2(c)) is proposed in [Song *et al.*, 2018]. The structure learns the side-outputs cumulatively, which progressively removes the detailed and noisy boundaries to enable high-resolution and accurate edge detection.

For bottom-up architecture, an ideal case is that the lower-level edge information is cumulatively inherited while the superfluous details are progressively abandoned. However, some useful low-level features are inevitably removed, which may cause uncorrectable effects to the resulting edge maps. This is the same case for the top-down architecture, which leads to sub-optimal results. So *why don't we make full use top-down and bottom-up architectures in a collaborative manner to complement the removed information and enhance feature hierarchy?* Based on that, we employ the progressive bottom-up and top-down branches to build a long-path information propagation network and refine the features in this consecutive information flow (Fig 2(d)). The combination of top-down and bottom-up architecture provides the network a chance of error-correcting, by revisiting the low-level features containing rich textual and spatial information. Furthermore, based on the observation that some features transmit superfluous or even wrong information to others as Fig 3, we incorporate channel attention mechanism into our network to select task-relevant and pivotal features.

In this paper, we propose an effective and end-to-end framework, named Bidirectional Additive Net (BAN) for image edge detection. The contributions of this paper can be summarized as: 1) We propose a progressive bottom-up and top-down information flow model for edge detection, where the feature hierarchy can be enhanced and hierarchical representations can be adaptively combined in this progressive bottom-up and top-down schemes. This architecture provides the network whit a chance of error-correcting and improves the performance gradually; 2) we propose an attended additive module to adaptively attend to each channel of the passing features so that the informative and accurate feature maps can be preserved, and the superfluous feature maps can be removed to avoid the cumulative error. To the best of our knowledge, we are the first to employ the channel-wise attention mechanism for image edge detection on deep learning; and 3) experimental results show that our proposed method can improve the edge detection performance to new records and achieve state-of-the-art results on two public benchmarks: BSDS500 and NYUDv2. The ablation study also verifies the effect of each component.

2 Our Method

In this paper, we propose a novel edge detection architecture, which combines hierarchical feature maps adequately in progressive bottom-up and top-down branches. Given an input image $I \in \mathbb{R}^{H\times W\times 3}$. Our network aims to predict an accurate fused edge map $E_f \in \mathbb{R}^{H\times W\times 1}$. The framework of our architecture is shown in Fig. 4. It contains four major components: 1) pyramid feature extractor network, which extracts richer spatial features from input images by atrous pyramid architecture, 2) bottom-up branches, 3) top-down branches and 4) attended additive module (AAM). We next elaborate each of them in details.

2.1 Pyramid Feature Extractor Network

The pyramid feature extractor can be divided into two subsets: feature extractor and pyramid atrous convolution module. We denote the basic feature extractor as V, and it contains multiple convolution layers, with pooling and activation operations in five stages. Given an input image I, the basic feature extractor generates an edge feature $F_i \in \mathbb{R}^{h^i\times w^i\times C}(i = 1, 2, ..., L)$ in each stage followed by a 3×3 convolution for feature dimension reduction. L denotes the number of stages in basic feature extractor and i denotes the i-th stage.

After obtaining simplify features $F_i \in \mathbb{R}^{h^i\times w^i\times C}$, a series of atrous convolution layer are employed to generate multi-receptive-field edge features with the pyramid structure. We set the kernel of the convolution layers as $A_r \in \mathbb{R}^{3\times 3\times C}(r = r_1, r_2, ..., r_K)$ which C denotes the output channel, r denotes

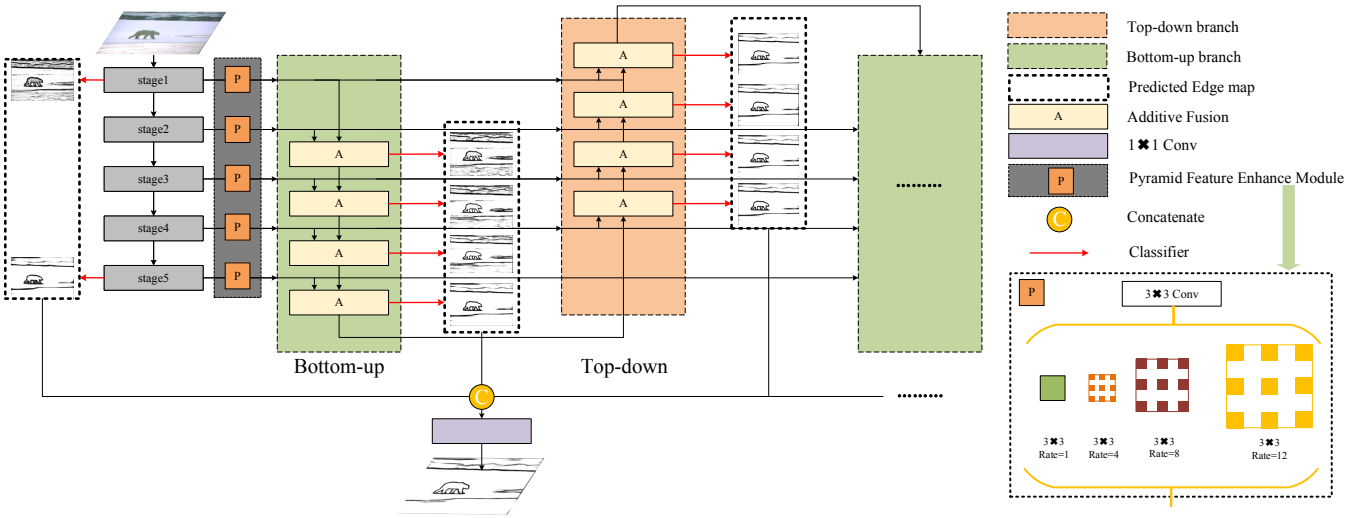


Figure 4: The framework of our proposed network. The input is an original image with arbitrary sizes, and the output of the network is an edge probability map with the same size. The feature extractor network employs the popular VGG-16 network.

the atrous rate and K denotes the number of atrous convolutions. We believe that resolution is very important in edge detection so we resize the features into input scales. This pyramid atrous convolutional operation is defined as:

$$P_i = U(\sum_{r=r_1}^{r_k}(A_r * F_i) + F_i) \tag{1}$$

where $*$ denotes the atrous convolution operation, P_i indicates the generated feature maps and U means the upsampling operations. We also add a residual shortcut between P_i and F_i in order to help gradients back propagation.

2.2 Bottom-up and Top-down Branches

Given an input image I, we generate enhanced edge features $P_i \in \mathbb{R}^{H\times W\times C}$ by pyramid feature extractor in Section 2.1. Then we employ successive bottom-up and top-down branches to refine the generated edge features. The AAM in this subsection represents the attended additive module, which will be elaborated in the next subsection.

Bottom-up branch. Given a set of pyramid enhanced edge features P_i, a bottom-up branch combines the lower level features with higher level features gradually and generates fused features $BU_i \in \mathbb{R}^{H\times W\times C}(i = 1, 2, .., L-1)$. For a lower level feature P_{i-1} and a higher level feature P_i, we can generate fused features by:

$$BU_i = \begin{cases} AAM(P_{i+1}, P_i), & \text{if } i = 1 \\ AAM(P_{i+1}, BU_{i-1}), & \text{if } i = 2, 3, ..., L-1 \end{cases} \tag{2}$$

Specially, we can observe that the final feature map carries richer information than others in a bottom-up branch. The experiments that is shown on Table 4 prove our conclusion and the effect of the bottom-up branch.

Top-down branch. Given a set of pyramid enhanced edge features P_i, a top-down branch generates features by transmitting upper-layer edge features to inferior edge features progressively. For an upper-layer feature map P_i and a lower feature map P_{i-1}, the top-down branch will generate features map $TD_i \in \mathbb{R}^{H\times W\times C}(i = L-1, L-2, ..., 1)$ as:

$$TD_i = \begin{cases} AAM(P_i, P_{i+1}), & \text{if } i = L-1 \\ AAM(P_i, TD_{i+1}), & \text{if } i = L-2, L-3, ..., 1 \end{cases} \tag{3}$$

Multi branches. In this subsection, we describe the progressive bottom-up and top-down branches structure. Following this inference scheme, TD_i^n denotes the fused feature map in the i-th stage from the top-down branch at n-th branch. In a similar way, we can define the output feature map in the i-th stage from the bottom-up and top-down branches at n-th branch as BU_i^n, $TD_i^n(n = 1, 2, ..., N, i = 1, 2, ..., L-1)$. The total number of branches is defined as N. The outputs of the corresponding bottom-up branch can be generated by:

$$BU_i^n = \begin{cases} AAM(P_{i+1}, TD_1^{n-1}), & \text{if } i = 1 \text{ and } n \geq 2 \\ AAM(P_{i+1}, P_i), & \text{if } i = 1 \text{ and } n = 1 \\ AAM(P_{i+1}, BU_{i-1}^n), & \text{if } i = 2, 3, ..., L-1 \end{cases} \tag{4}$$

The generated features from corresponding top-down branch are represented as:

$$TD_i^n = \begin{cases} AAM(P_i, BU_{L-1}^{n-1}), & \text{if } i = L-1 \text{ and } n \geq 2 \\ AAM(P_i, P_{i+1}), & \text{if } i = L-1 \text{ and } n = 1 \\ AAM(P_i, TD_{i+1}^n), & \text{if } i = L-2, L-3, ..., 1 \end{cases} \tag{5}$$

2.3 Attended Additive Module

After employing the progressive bottom-up and top-down branches into our network, we can utilize multi-level features for the generation of image edge maps. In this subsection, we introduce the details of Attended Additive Module for probably fusing these features. The structure is shown in Fig 5.

In general, let $f \in \mathbb{R}^{H\times W\times C}$ denote an arbitrary flowing feature map that has C channels. Firstly, we apply pooling

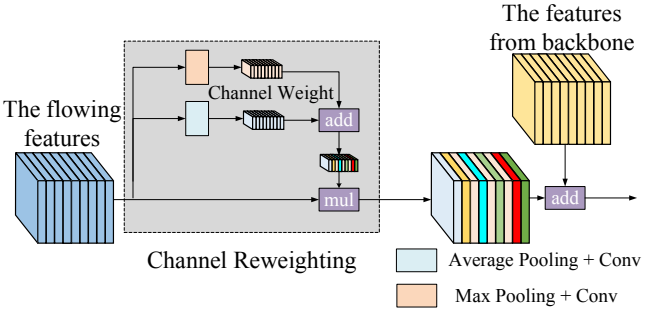


Figure 5: The detailed structure of Attended Additive Module

layer to f to obtain a vector $v \in \mathbb{R}^C$, which contains the global information of f. Then two progressive $1\times$ 1 convolution layers (the same as fully connected layer here) are employed to investigate the importance of each channel and the relationship between themselves. A sigmoid operation is employed to normalize the weight to $[0, 1]$. This process can be represented by the following equation:

$$w = Sigmoid(Conv(\sigma(Conv(Pool(f))))) \tag{6}$$

where w denotes the channel weight, σ represents the ReLU activation function, $Conv$ refers to convolution layer and $Pool$ refers to pooling layer. So the $AAM(\cdot, \cdot)$ in last subsection is defined as:

$$AAM(P_i, f) = w * f + P_i \tag{7}$$

Inspired by [Bertasius *et al.*, 2015], we employ two different pooling schemes, e.g. average and max pooling, for channel reweighting.

2.4 Network Optimization

Loss function. The original cross entropy loss cannot get satisfactory performance because of the unbalance between positive and negative classes for edge detection. Following the previous work [Liu *et al.*, 2019] [He *et al.*, 2019], we employ a class-balanced cross entropy loss function to train the network. The loss function is defined as:

$$\begin{aligned} L_{ce}(W) = &-\gamma \cdot \beta \sum\nolimits_{g_i \in G_+} \log P(g_i = 1|I; W) \\ &-(1-\beta) \sum\nolimits_{g_i \in G_-} \log P(g_i = 0|I; W) \end{aligned} \tag{8}$$

where I is an input image, and G is the final ground truth of the input image I. G_+ denotes all of the edge pixels, while G_- is all of the non-edge pixels. $\beta = \frac{|G_-|}{|G|}$ is the ratio of edge pixels in all pixels, which is used to reweigh the positive and negative classes weights when computing loss. γ is a hyper-parameter to balance the edge and non-edge pixels ratio. W refers to all trainable parameters in our network. But the network can't generate crisp edge maps on the condition of only cross entropy loss without the non-maximum suppression (NMS). Follow the previous work, we also employ dice loss to thin the predicted edges. The e_i denotes a pixel in predicted edge maps.

$$L_d(W) = \frac{\sum_i^N e_i^2 + \sum_i^N g_i^2}{2\sum_i^N e_i g_i} \tag{9}$$

The mixed loss function is defined as :

$$L(W) = L_{ce}(W) + \lambda_d L_d(W) \tag{10}$$

where λ_d is a hyper-parameter that controls the trade-off between cross-entropy loss and dice loss.

Training strategies. To obtain a final edge map, we first generate a weighted-fusion output E_{fuse} by a weighted linear combination of side-outputs $E_i^n (i = 1, 2, ..., L; n = 1, 2, ..., N)$. E_i^n is generated by each Attended Additive Module followed by a $1 \times 1 \times 1$ convolutional layer as shown in Fig 4. Inspired by the [Xie and Tu, 2017], we merge the weighted-fusion output E_{fuse} with the side-output E_i^n. We also supplement the edge maps generated by F_1 and F_5 to the final output for obtaining sufficient details and semantic information. We employ a deep supervision training strategy to train our network. It can restrain the gradients vanishing problem and guarantee a satisfactory edge prediction in each side-output, which is helpful to get a final prediction.

3 Experiments

3.1 Experiments Setup

Datasets. To evaluate our proposed method, we use two commonly used benchmark datasets: BSDS500 [Arbelaez *et al.*, 2011] and NYUDv2 [Silberman *et al.*, 2012]. BSDS500 benchmark dataset is an extension of the BSDS300. Follow [Xie and Tu, 2017] [Liu *et al.*, 2019], we train our network on training and validation sets and test our method on test set. NYUDv2 is collected from a variety of indoor scenes with 1,449 RGB images and depth images. We split the whole images into two subsets: 756 for training and 654 for testing. To prevent over-fitting, we apply data augmentation strategies following [Xie and Tu, 2017].

Evaluation metrics. Following the existing works, the non-maximum suppression (NMS) is employed to generate thinned edge maps firstly. After that, we use F-measures at optimal dataset scale (ODS), optical image scale (OIS) and average precision(AP) for evaluation metrics. In the matching process, we set the maximum tolerance distance to 0.0075 for BSDS500 and 0.011 for NYUDv2 respectively.

Implementation details. We initialize our network by the VGG16 model pre-trained on ImageNet. We set the stride of pool4 to 1 and apply atrous convolution in the last stage of VGG16 to obtain reasonable resolution, which is very important for dense pixel classification. The *Fuse* operations in our network are implemented by addition operations. The convolution layer in our network are initialized from zero-mean Gaussian distribution with 0.01 standard deviation. The hyper-parameter λ_d in Equation 10 is set as 300. The C, N and L are set to 21, 2 and 5 in Section 2. Following [Liu *et al.*, 2019], λ is set to 1.1 for BSDS500 and 1.2 for NYUDv2 respectively. We employ SGD optimization during training. The learning rate, weight decay, momentum and batch size are set to 1e-7, 0.9, 2e-4 and 10, respectively. We train BSDS500 for 15k steps and NYUDv2 for 30k steps because of the different sizes of training set.

Variants	ODS	OIS	AP
Baseline	0.794	0.812	0.779
n=1(v1)	0.800	0.818	0.809
n=2(v1)	**0.810**	**0.827**	0.862
n=3(v1)	0.806	0.824	0.847
n=1(v2)	0.801	0.819	0.830
n=2(v2)	0.808	0.825	0.864
n=3(v2)	0.805	0.822	0.839

Table 1: The influence of number of branches and comparison between two variants architectures. Other settings are fixed.

Methods	ODS	OIS	AP
Baseline	0.794	0.812	0.779
Prediction-level	0.797	0.815	0.837
Feature-level	0.805	0.821	0.840
AAM(average)	0.809	0.826	0.848
AAM(max)	0.809	0.825	0.840
AAM(two-stream)	**0.810**	**0.827**	**0.862**

Table 2: Ablation study for different information flow mechanism and the attended additive model. The number of top-down and bottom-up branches is set to 2. 'Prediction-level' and 'Feature-level' represent directly fusing the edge maps and feature maps. 'average' and 'max' mean average and max pooling layer. 'two-steam' indicates that employing two different pooling layers.

3.2 Ablation Study

The *Baseline* in ablation study means that the five side-outputs are fused by a concatenation operation followed by an $1 \times 1 \times 1$ convolution layer without information flow. We conduct experiments using BSDS500 dataset.

Different variants. Firstly we investigate the performance of different variants. We discuss two variants here: 1) first bottom-up branch and then top-down branch, 2) first top-down branch and then bottom-up branch. We call the first variant as *'v1'* and another one *'v2'*. We evaluate the performance of two variants and choose the 'v1' variant as our structure based on the experimental results. From Table1, we can observe that both *'v1'* and *'v2'* outperform the baseline, which shows the effectiveness and importance of information flow between different stages.

Different numbers of bottom-up and top-down branches. From Table 1, we can observe that the multiply top-down and bottom-up branches enhance the performance of edge detection greatly (n$\leq$2). But when the n=3, the performance declines slightly. The reason might be that the too deep network makes the training difficult, especially for the edge detection which has extremely unbalanced and scarce training data. Even so, the performances are also outperform the baseline and single direction (n=1).

Training Strategies	ODS	OIS	AP
network(w/o deep supervision)	0.796	0.814	0.784
network(only dice loss)	0.790	0.807	0.827
network(w/o dice loss)	0.809	0.827	0.860
network(w/ dice loss)	0.810	0.827	0.862

Table 3: Ablation study for different loss function and training strategy. The network is our proposed network.

Side-outputs	ODS		
	Baseline	T=1	T=2
stage1	0.722	0.743	0.804
stage2	0.749	0.770	0.803
stage3	0.771	0.788	0.803
stage4	0.756	0.802	0.804
stage5	0.757	0.803	-
fuse	0.794	**0.810**	

Table 4: The comparison of side-outputs between baseline and our network. We compared the performance on ODS, which is the most important evaluation metrics in edge detection.

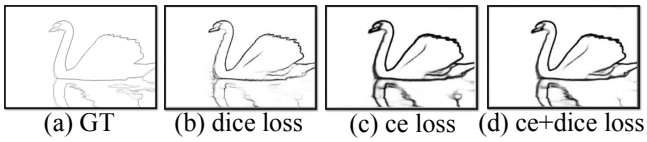


Figure 6: The qualitative comparison of different different loss function. 'ce loss' denotes weighted binary cross entropy loss.

Prediction-level vs. Feature-level. One of the differences between our method and [He *et al.*, 2019] is that we employ a feature-level information flow model rather than prediction-level. Because we think the feature maps contain richer information than edge maps. In Table 2, we compare the effect of prediction-level and feature-level information flow. From Table 2, We can draw the following conclusions: (1) The prediction-level information flow is useful for edge detection, which increases all evaluation metrics by 0.3% ODS,0.3% and 5.8% AP; (2) feature-level information flow improves the performance of edge detection significantly. Compared with the baseline, feature-level information flow performs it by 1.1% ODS, 0.9% OIS and 6.1% AP, which has greater improvement than prediction-level information flow. These prove the effectiveness of feature-level information flow mechanism and progressive bottom-up and top-down branches.

Attended additive module and different pooling schemes. In Table 2, we also discuss the attended additive module and the different pooling schemes in the module. The *Feature-level* denotes that AAM without channel reweighting mechanism. As can be seen, the channel reweighting mechanism in AAM improves the performance by filtering unnecessary and noisy information and select important components on flowing features. And the different pooling schemes also improve the performance of edge detection slightly. The reason might be that the different pooling schemes capture different global information in channel reweighting, which makes the reweighting operations more robust and effective.

Side-outputs. As shown in Table 4, different stages in our networks gradually produce better performance, while the final results fuse all the side-outputs and obtain the best performance. This indicates the effect of our proposed architecture.

The loss function and training strategy. Our model is trained by a fused loss function with a deep supervision strategy. The supervisions are fed to each side-output in each branch. Tab. 3 shows that deep supervision is helpful for edge

Methods	ODS	OIS	AP
Human	0.803	0.803	-
Canny	0.611	0.67	0.520
Pb	0.672	0.695	0.652
MCG	0.744	0.777	0.760
OEF	0.746	0.770	0.815
DeepEdge	0.753	0.772	0.807
DeepContour	0.757	0.776	0.790
HED	0.788	0.808	0.840
COB	0.793	0.819	0.849
RCF	0.798	0.815	-
LPCB	0.800	0.816	-
CED	0.794	0.811	0.847
CED-MS	0.803	0.820	0.871
BDCN	0.806	0.826	0.847
AMH-Net(ResNet50)	0.798	0.829	0.869
CNet(ResNet101)	0.805	0.819	0.851
BAN	**0.810**	**0.827**	**0.862**
BAN-MS	**0.816**	**0.834**	**0.870**

Table 5: The comparison with other methods on BSDS500 dataset.'MS' in the table indicates the results of multi-scale testing

Methods	ODS	OIS	AP
gPb-UCM	0.631	0.661	0.562
OEF	0.651	0.667	0.653
gPb+NG	0.687	0.716	0.629
SE	0.695	0.708	0.719
SE+NG+	0.706	0.734	0.549
HED-RGB	0.720	0.734	0.734
HED-HHA	0.682	0.695	0.702
HED-RGB-HHA	0.746	0.761	0.786
RCF-RGB	0.729	0.742	-
RCF-HHA	0.705	0.715	-
RCF-RGB-HHA	0.757	0.771	-
LPCB-RGB	0.739	0.754	-
LPCB-HHA	0.707	0.719	-
LPCB-RGB-HHA	0.762	0.778	-
BDCN-RGB	0.748	0.763	0.770
BDCN-HHA	0.707	0.719	0.731
BDCN-RGB-HHA	0.765	0.781	0.813
AMH-Net(ResNet50)-RGB-HHA	0.771	0.786	0.802
CNet(ResNet101)-RGB-HHA	0.762	0.781	0.797
BAN-RGB	**0.755**	**0.770**	0.760
BAN-HHA	0.707	0.718	0.710
BAN-RGB-HHA	**0.773**	**0.786**	0.790

Table 6: The comparison with other methods on NYUDv2 dataset.

detection. The network trained by fused loss function obtains a slightly better performance. And the dice loss makes the edges crisp, which is helpful for other tasks such as optical flow and object proposal, as shown in Fig. 6.

3.3 Comparison with State-of-the-art

Comparison on BSDS500. In this subsection, we perform comparisons of our proposed BAN and other methods, including traditional methods (Canny [Canny, 1986], Pb [Martin *et al.*, 2004], MCG [Pont-Tuset *et al.*, 2017] and OEF [Hallman and Fowlkes, 2015]) and deep learning methods (DeepEdge [Bertasius *et al.*, 2015], DeepContour [Shen *et al.*, 2015b], HED [Xie and Tu, 2017], COB [Maninis *et al.*, 2018], RCF [Liu *et al.*, 2019], LPCB [Wang *et al.*, 2019b], BDCN[He *et al.*, 2019], AMH-Net [Xu *et al.*, 2017] and CNet [Song *et al.*, 2018]). The experimental results are summarized in Table 5. From Table 5 we can observe that our method obtains F-measure of 0.810 on ODS in single-scale input images and 0.816 of multi-scale input images, which outperforms all the previous approaches. ODS is the most important evaluation metric in edge detection which reflects the performance on the whole dataset. BAN also obtains better performance compared with CED-MS that uses multi-scale input images and ResNet-based methods such as AMH-Net and CNet on ODS. We observe that our method outperforms the others, including human for all evaluation metrics.

Comparison on NYUDv2. In this subsection, we compare our method with state of the art methods, including gpb-UCM [Arbelaez *et al.*, 2011], OEF [Hallman and Fowlkes, 2015], gpb+NG [Gupta *et al.*, 2013], SE [Dollár and Zitnick, 2015], SE+NG+ [Gupta *et al.*, 2014], HED [Xie and Tu, 2017], RCF [Liu *et al.*, 2019], LPCB [Deng *et al.*, 2018] and BDCN [He *et al.*, 2019]. In the recent work such as [Xie and Tu, 2017] and [Liu *et al.*, 2019], the final result are obtained by averaging the results from two seperately models which are trained by RGB images and HHA images respectively. The comparison results are shown in Table 6. As shown in the results, our method outperforms other recent approaches including some ResNet-based approaches. Specifically, we can see that our method performs better than the best VGG-based approach BDCN(RGB) with an increase of 0.7% ODS and 0.7% OIS. When we combine RGB and HHA, the performance of our method sets a net record and reaches 0.773, 0.786 and 0.790. It's worth noting that the F-measure at ODS and OIS are more important evaluation metrics than AP for edge detection [Liu *et al.*, 2019] [Deng *et al.*, 2018]. From Table 6, we can also notice that our method outperforms ResNet-based methods: CNet and AMH-Net at F-measures.

4 Conclusion

In this paper, we propose a novel progressive bottom-up and top-down information flow model named Bidirectional Additive Net (BAN), which utilizes the feature-level information flow from bottom-up to top-down branches to enhance information propagation and obtain richer feature hierarchy. We filter the propagating information by reweighting each channel of feature maps by Attended Additive Module (AAM). Experimental results show that our proposed method outperforms the state-of-the-art on two benchmark datasets.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61772116, No. 61872064, No.61632007, No. 61602049), The Open Project of Zhejiang Lab (Grant No.2019KD0AB05), Huo Yingdong Education Foundation and Sichuan Science and Technology Program (No. 2018GZDZX0032 and No. 2019JDTD0005).

References

[Arbelaez *et al.*, 2011] Pablo Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.

[Bertasius *et al.*, 2015] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated

deep network for top-down contour detection. In *CVPR*, pages 4380–4389, 2015.

[Canny, 1986] John F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[Chen *et al.*, 2016] Liang-Chieh Chen, Jonathan T. Barron, George Papandreou, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *CVPR*, pages 4545–4554, 2016.

[Deng *et al.*, 2018] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *ECCV*, pages 570–586, 2018.

[Dollár and Zitnick, 2015] Piotr Dollár and C. Lawrence Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2015.

[Gao *et al.*, 2019] Lianli Gao, Xiangpeng Li, Jingkuan Song, and Heng Tao Shen. Hierarchical lstms with adaptive attention for visual captioning. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[Guo *et al.*, 2019] Yuyu Guo, Lianli Gao, Jingkuan Song, Peng Wang, Wuyuan Xie, and Heng Tao Shen. Adaptive multi-path aggregation for human densepose estimation in the wild. In *ACM Multimedia*, pages 356–364, 2019.

[Gupta *et al.*, 2013] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, pages 564–571, 2013.

[Gupta *et al.*, 2014] Saurabh Gupta, Ross B. Girshick, Pablo Andrés Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, pages 345–360, 2014.

[Hallman and Fowlkes, 2015] Sam Hallman and Charless C. Fowlkes. Oriented edge forests for boundary detection. In *CVPR*, pages 1732–1740, 2015.

[He *et al.*, 2019] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *CVPR*, pages 3828–3837, 2019.

[Liu *et al.*, 2019] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Jia-Wang Bian, Le Zhang, Xiang Bai, and Jinhui Tang. Richer convolutional features for edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1939–1946, 2019.

[Maninis *et al.*, 2018] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbelaez, and Luc Van Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):819–833, 2018.

[Martin *et al.*, 2004] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004.

[Pont-Tuset *et al.*, 2017] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T. Barron, Ferran Marqués, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(1):128–140, 2017.

[Shen *et al.*, 2015a] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, Zhenmin Tang, and Heng Tao Shen. Hashing on nonlinear manifolds. *IEEE Trans. Image Processing*, 24(6):1839–1851, 2015.

[Shen *et al.*, 2015b] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, pages 3982–3991, 2015.

[Silberman *et al.*, 2012] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, pages 746–760, 2012.

[Song *et al.*, 2018] Jingkuan Song, Zhilong Zhou, Lianli Gao, Xing Xu, and Heng Tao Shen. Cumulative nets for edge detection. In *ACM MM*, pages 1847–1855, 2018.

[Wang *et al.*, 2019a] Wenguan Wang, Shuyang Zhao, Jianbing Shen, Steven C. H. Hoi, and Ali Borji. Salient object detection with pyramid attention and salient edges. In *CVPR*, pages 1448–1457, 2019.

[Wang *et al.*, 2019b] Yupei Wang, Xin Zhao, Yin Li, and Kaiqi Huang. Deep crisp boundaries: From boundaries to higher-level tasks. *IEEE Trans. Image Processing*, 28(3):1285–1298, 2019.

[Xie and Tu, 2017] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *International Journal of Computer Vision*, 125(1-3):3–18, 2017.

[Xu *et al.*, 2017] Dan Xu, Wanli Ouyang, Xavier Alameda-Pineda, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. Learning deep structured multi-scale features using attention-gated crfs for contour prediction. In *NeurIPS*, pages 3961–3970, 2017.

[Yang *et al.*, 2016] Jimei Yang, Brian L. Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, pages 193–202, 2016.

[Zhu *et al.*, 2016] Xiaofeng Zhu, Xuelong Li, Shichao Zhang, Chunhua Ju, and Xindong Wu. Robust joint graph sparse coding for unsupervised spectral feature selection. *IEEE transactions on neural networks and learning systems*, 28(6):1263–1275, 2016.