Article

Adaptive Feature Pyramid Network to Predict Crisp Boundaries via NMS Layer and ODS F-Measure Loss Function

Gang Sun, Hancheng Yu *, Xiangtao Jiang and Mingkui Feng

College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; sungang@nuaa.edu.cn (G.S.); jxt105671@nuaa.edu.cn (X.J.); mkfeng@nuaa.edu.cn (M.F.)
* Correspondence: yuhc@nuaa.edu.cn

Abstract: Edge detection is one of the fundamental computer vision tasks. Recent methods for edge detection based on a convolutional neural network (CNN) typically employ the weighted cross-entropy loss. Their predicted results being thick and needing post-processing before calculating the optimal dataset scale (ODS) F-measure for evaluation. To achieve end-to-end training, we propose a non-maximum suppression layer (NMS) to obtain sharp boundaries without the need for post-processing. The ODS F-measure can be calculated based on these sharp boundaries. So, the ODS F-measure loss function is proposed to train the network. Besides, we propose an adaptive multi-level feature pyramid network (AFPN) to better fuse different levels of features. Furthermore, to enrich multi-scale features learned by AFPN, we introduce a pyramid context module (PCM) that includes dilated convolution to extract multi-scale features. Experimental results indicate that the proposed AFPN achieves state-of-the-art performance on the BSDS500 dataset (ODS F-score of 0.837) and the NYUDv2 dataset (ODS F-score of 0.780).

Keywords: edge detection; convolution neural network; non-maximum suppression layer; ODS F-measure loss function; multi-level feature pyramid network

Citation: Sun, G.; Yu, H.; Jiang, X.; Feng, M. Adaptive Feature Pyramid Network to Predict Crisp Boundaries via NMS Layer and ODS F-Measure Loss Function. *Information* **2022**, *13*, 32. https://doi.org/10.3390/info13010032

Academic Editor: Francesco Beritelli

Received: 23 November 2021
Accepted: 25 December 2021
Published: 12 January 2022

1. Introduction

Edge detection is one of the fundamental computer vision tasks, which plays a significant role in many challenging computer vision applications such as object recognition [1,2], image segmentation [3,4], and object proposal [5]. Psychological studies suggest that when a human pinpoints boundaries in natural images, object-level reasoning is used. So, edges and boundaries are usually defined as semantically meaningful.

Nevertheless, early works such as the Sobel detector [6] and the Canny detector [7] extract some local low-level features, and it is difficult to use these low-level clues to represent object-level information. In this case, sketch tokens [8] and structured edges [9] tried to use complex strategies to capture as many global features as possible, making some progress. Recently, relying on the advances in deep learning, edge detection has been substantially improved. CNN-based methods, such as holistically-nested edge detection (HED) [10] and richer convolution features for edge detection (RCF) [11], take advantage of hierarchical feature learning and have demonstrated a state-of-the-art F-score performance on the BSDS500.

As shown in Figure 1, compared with the Canny detector, HED can make the edges more continuous and remove unimportant edges. However, the results of HED are much thicker than the results of the Canny detector. Deng et al. [12] proved that the weighted cross-entropy loss of the detector is the main cause of the thick edge.

Edge detection performance commonly uses optimal dataset scale (ODS) F-measure as evaluation metrics. To achieve end-to-end training, we propose a non-maximum suppression layer (NMS) to obtain a sharp boundary, which can be used to calculated the ODS F-measure directly. Then, the ODS F-measure loss is proposed to train the network, which can remove noise and suppress unimportant edges more effectively.

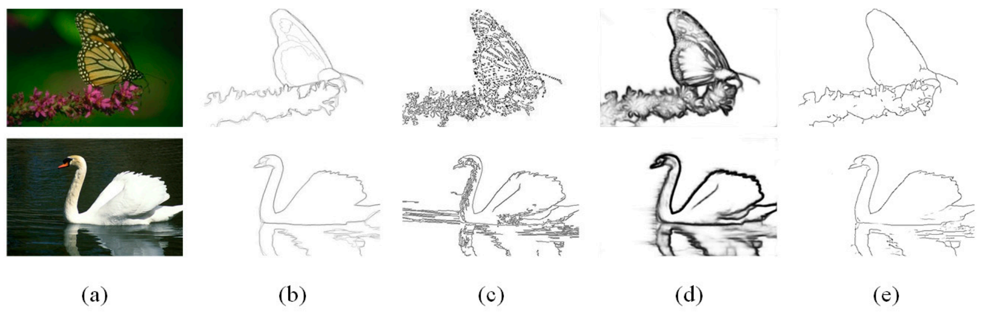

Figure 1. (**a**) Are example images from the BSDS500 dataset; (**b**) are the ground truths; (**c**) are the results of the Canny detector; (**d**) are the outputs of the HED detector; and (**e**) are the outputs of our proposed method. All the results do not apply post-processing.

The boundary ratio of the object in the natural image varies greatly, not only containing the overall boundary information, but also some local detail information. How to extract this multi-scale information is particularly important for boundary detection. HED and RCF use VGG [13] as the backbone network to extract feature information of different scales in the image. They use multi-level side outputs architecture and train these levels with the same supervision. It is not optimal, since different network levels depict patterns at different scales. An FPN (feature pyramid network) [14] is employed in numerous methods to alleviate the problems caused by large scale range of objects, such as Wang et al. [15] using FPN for object detection. We use FPN for detecting the edges of different objects and propose an adaptive multi-level pyramid network, where each layer is supervised by the labeled edges at its specific scale, rather than directly applying the same supervision to all levels. Aiming to fully exploit the multi-scale features with a shallow CNN, we introduce a pyramid context module (PCM) that consists of multiple parallel convolutions with different dilation rates. By involving multiple dilated convolutions, PCM captures multi-scale spatial contexts and does not significantly increase network parameters.

This paper proposes an end-to-end complete convolutional network for accurate image boundary prediction. Figure 1 shows two examples of the edge quality improvement between our method and the HED detector. The major contributions of this paper are:

- We propose a non-maximum suppression layer (NMS) to obtain a sharp boundary directly without the need for post-processing.
- Secondly, the ODS F-measure loss function is proposed to achieve end-to-end boundary detection training.
- Lastly, we propose an adaptive multi-level feature pyramid network (AFPN) to better combine different levels of features.

2. Related Work

The edge detection algorithm is the preprocessing process of many advanced tasks. So far, many researchers have been studying the problem continuously and have made remarkable achievements. Reviewing the development process of edge extraction algorithms, we can roughly categorize these existing edge detection algorithms into three groups: traditional edge operators, learning-based methods, and recent deep learning methods.

Early research focused on designing simple filters to detect pixels with the highest gradients in their local neighborhoods, such as the Sobel operator and the Canny operator. Filter-based methods can produce better detection results, by providing more accurate edge information, plus they have a soothing effect on noise. Learning-based methods usually integrate different low-level features and train a classifier to generate object-level contours. SE (Structured Edges) uses the structure of local image patches to learn an accurate and computationally efficient edge detector. Structured random forests (SRF) [16] use the semi-supervised learning method for edge detection for the first time, proposing a semi-supervised structured ensemble learning method for contour detection based on the structured random forest.

Compared with the earlier work, the above methods have made great progress. However, these methods are all developed based on manual features and cannot express the high-level information of semantic edge detection. With the development of deep learning, researchers have invented a series of edge detection methods based on deep learning. Ganin et al. [17] proposed N^4-fields, which combines deep neural networks and nearest neighbor search to obtain edge detection results. Then, Bertasius et al. [18] expanded N^4-fields and proposed the DeepEdge, which is a unified multi-scale deep learning method. It first uses the Canny operator to extract candidate edge points and then uses the deep neural network KNet to extract high-level features of the picture. Shen et al. [19] proposed DeepContour, in which edge data are divided into several different sub-categories. It uses the method of model parameter fitting to learn each sub-category to obtain the final result.

The HED uses VGG16 as the basic network structure with some modifications. When evaluating on the well-known BSDS500 [3] benchmark, it achieves the ODS F-measure of 0.782, which significantly surpasses the most advanced technology at the time. HED uses multi-level convolution and down-sampling to learn rich multi-scale information and uses multi-level side outputs to learn rich hierarchical information. It is of great significance for solving the challenging ambiguity in edge and target boundary detection. Finally, the purpose of accurately positioning the edge is achieved by fusing multiple layers of information. RCF believes that objects in natural images have different scales and aspect ratios. Learning-rich hierarchical representations are essential for edge detection. The network makes full use of the multi-scale and multi-level information of the objects. It also performs image-to-image prediction, by integrating all meaningful convolution features. BDCN [20] proposes a bi-directional cascade network structure, in which a single layer is supervised by labeled edges with a specific scale, instead of directly applying the same supervision to all CNN outputs. Additionally, it adds an SEM (Scale Enhancement Module) to the network, which is composed of multiple parallel convolutions with different expansion rates to generate multi-scale feature maps. The dilated convolution effectively increases the size of the receptive field of the network neurons. It does not significantly increase the network parameters and avoids repeated edge detection of the image pyramid.

Although these aforementioned CNN-based models can improve the edge detection performance significantly, they typically suffer from the issue of predicted edges being thick and needing post-processing before evaluation. To solve these problems, we propose a non-maximum suppression layer, the ODS F-measure loss function, and a top-down adaptive multi-level feature pyramid network. Next, we will introduce our method in detail.

3. Approach

3.1. Architecture

As shown in Figure 2, adaptive feature pyramid network architecture is proposed for edge detection. The pyramid structure, from high-level features to low-level features, preserves detailed features while extracting object-level features. Aiming to fully exploit the multi-scale features with a shallow CNN, we introduce a pyramid context module (PCM), which consists of multiple parallel convolutions with different dilation rates. By involving multiple dilated convolutions, PCM captures multi-scale spatial contexts and does not significantly increase network parameters.

The backbone of the network is based on VGG16, by removing three fully connected layers and the last pooling layer. The modified VGG16 has 13 convolutional layers, which are divided into 5 blocks, each follows a pooling layer to progressively enlarge the receptive fields in the next block. The network takes image $I \in \mathbb{R}^{3\times H\times W}$ with height H and width W as input. We denote the output feature maps of blocks in VGG16 as $R_t \in \mathbb{R}^{C\times \frac{H}{m}\times\frac{W}{m}}$, where t is the number of the block, C is the number of channels, and m is the stride of VGG16. This PCM takes the outputs of the blocks of VGG16 as input through a 1×1 convolution. We denoted the output of each PCM as $G_k \in \mathbb{R}^{C\times \frac{H}{m}\times\frac{W}{m}}$, where k is the number of PCM. When $k = 4$, the high level feature map is $\text{Conv}_{1\times1}(R_5)$ and the low-level feature map is

$\mathrm{Conv}_{1\times1}(R_4)$. Otherwise, the high-level feature map is G_{k+1}, and the low-level feature map is $\mathrm{Conv}_{1\times1}(R_k)$.

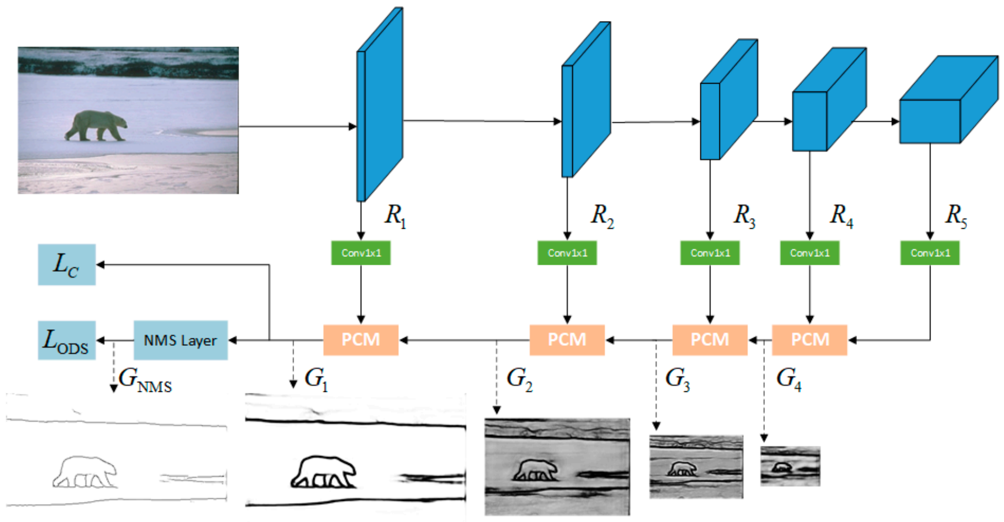

Figure 2. Adaptive feature pyramid network architecture.

The output results of each PCM are shown in Figure 2. In the deep layer of the network, the larger object boundary obtains a strong response. With the increase in levels, the deep high-level semantic information gradually merges with the shallow spatial information, and the edges are gradually clear. Pyramid context module (PCM) is a core component of our network, which utilizes dilated convolution to generate multi-scale features.

The architecture of PCM is shown in Figure 3. The high feature map is upsampled by deconvolution, to make it the same size as the low feature map and then concatenate the two feature maps. Inspired by ASPP [21], several dilated convolutions with different expansion rates are used in parallel to achieve multi-scale feature.

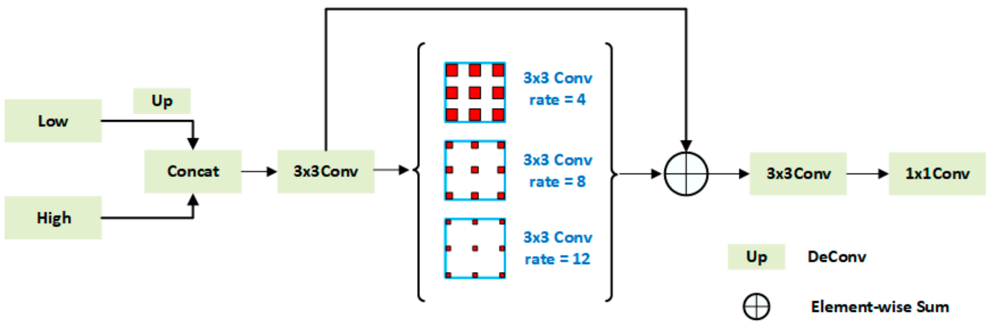

Figure 3. PCM architecture.

3.2. Non-Maximum Suppression Layer

Almost all recent CNNs-based methods suffer from the issue of predicted boundaries being thick. To obtain sharp boundaries, these methods use non-maximum suppression as post-processing. In this paper, a non-maximum suppression layer is proposed to obtain sharp boundaries without post-processing, and our network can directly output sharp boundaries. Deng et al. [12] proved that directly making sharp boundaries not only promotes the visual results of CNNs but also achieves better performance on several available datasets through experiments. Adding the non-maximum suppression layer only increases a small amount of calculation. In the proposed non-maximum suppression layer, the direction of the input gradient magnitudes G is defined as:

$$\theta = \arctan\left(\frac{G_{yy}\,\mathrm{sign}(G_{xy})}{G_{xx}}\right) \tag{1}$$

where G_{xx}, G_{yy}, and G_{xy} are the second derivative of G in the x direction, y direction, and diagonal direction, respectively, sign () is symbolic function. Then, we define the non-maximum suppression function:

$$G_{\mathrm{NMS}} = f(G) = \begin{cases} G \text{ if } G \text{ is local max in its gradient direction} \\ 0 \text{ else } G \text{ should be suppressed} \end{cases} \tag{2}$$

As shown in Figure 2, compared with the input gradient magnitudes G_1, the output boundaries of the non-maximum suppression layer is sharper.

Non-maximum suppression layer can be trained using backpropagation with other layers. The gradient of this layer is given by:

$$\frac{\partial f(G)}{\partial G} = \begin{cases} 1 \text{ if } G \text{ is local max in its gradient direction} \\ \alpha \text{ otherwise} \end{cases} \tag{3}$$

where α is a coefficient controlling the negative non-maximum gradient magnitudes. In our experiments, if α is set to 0, the non-maximum suppression layer will suppress some useful boundaries in training. In this paper, according to the experimental results, α is set to 0.4.

3.3. ODS F-Measure Loss

F-measure is used as the evaluation standard when the current edge detection results are evaluated on the well-known BSDS500 benchmark. It is defined as:

$$F = \frac{2 \times P \times R}{P + R} \tag{4}$$

P and R denote precision and recall, respectively:

$$\begin{cases} P = \dfrac{\sum\limits_k \sum\limits_i G^{ex} \times L_k}{\sum\limits_k \sum\limits_i G_k} \\ R = \dfrac{\sum\limits_k \sum\limits_i G \times L_k^{ex}}{\sum\limits_k \sum\limits_i L_k} \end{cases} \tag{5}$$

where the i denotes the i-th pixel and L_k denotes the k-th ground truth edge map labeled by multiple annotators. G^{ex} and L_k^{ex} denote the extended results of G and L_k, which allow d pixel error between the label and the predicted edges in calculating the precision and recall.

The precision-recall rate curve is a standard evaluation technique in the field of information retrieval [22]. It was first used by Abdul and Pratt [23] to evaluate edge detectors. Each point on the precision-recall rate curve is calculated based on the output under a certain threshold. There are two options for setting this threshold. The first is optimal dataset scale (ODS), which uses a fixed threshold for all images in the data set. The second is optimal image scale (OIS), which selects an optimal threshold for each image. Based on the sharp edge which obtained by the proposed non-maximum suppression layer, the ODS F-measure loss is defined as:

$$L_{\mathrm{ODS}}(G_{\mathrm{NMS}}, L) = 1 - \frac{2 \times P \times R}{P + R} \tag{6}$$

To achieve better performance, in this paper, the cross-entropy loss and the ODS F-measure loss are combined to train the network, as shown in Figure 2. The final loss function is defined as:

$$L = L_C(G_4, L) + \gamma L_{\mathrm{ODS}}(G_{\mathrm{NMS}}, L) \tag{7}$$

The hyper-parameter γ is to control the influence of two losses:

$$\gamma = \frac{N_{neg}}{\varepsilon} \tag{8}$$

where N_{neg} is the number of negative samples in the label. ε is a coefficient depending on the dataset, and we will discuss it in detail in the next experimental section.

4. Experiments

4.1. The BSDS500 and the NYUDv2 Dataset

The BSDS500 is a dataset widely used in edge detection, including 200 images for training, 100 images for verification, and 200 images for testing. Each image is manually annotated by multiple annotators. We follow the way of RCF to average all labels to generate a probability map and take a threshold value η. If it is greater than η, it is a positive sample, while if it is equal to 0, it is a negative sample, and the rest are ignored. Inspired by the previous work, we increase the training set by randomly flipping, scaling, and rotating the 300 labels obtained from training, and then mixing the obtained training set with the flipped PASCAL VOC dataset as training data.

The NYUDv2 [24] dataset consists of 1449 RGB-D images. The NYUDv2 was originally used for scene understanding, so it was also used for edge detection in previous work. Following the previous work, we performed the same data enhancement, rotated the image and corresponding annotations to four different angles (0 degrees, 90 degrees, 180 degrees, and 270 degrees), and flipped them at each angle. We trained two different models, one with RGB image as input, and the other with HHA image as input. Finally, the output of the RGB model and the HHA model is averaged as the final edge prediction result.

4.2. Implementation Details

We use the publicly available Pytorch [25] to implement our network and set the training parameters following previous work, such as HED and RCF. The default setting uses a VGG16 [13] backbone net, and we also test a ResNet [26] backbone net. The weights of the 1×1 Conv layer in blocks 1–5 are initialized from a zero-mean Gaussian distribution with a standard deviation of 0.01, and the deviation is initialized to 0. The weights of other layers are initialized using pre-trained ImageNet [27] models. SGD optimizer is adopted to train our network. The hyper-parameters of our model include: mini-batch size (10), learning rate (1×10^{-6}), momentum (0.9), weight decay (0.0002), and number of training iterations (10,000; divide learning rate by 10 after 5000). We run a total of 40 k SGD iterations. When training on the BSDS500 and the NYUDv2, the pixel error d in Equation (5) is set to 3 and 4, ε in Equation (8) is set to 300 and 2000, respectively, according to the experimental results. All experiments in this paper are performed on GTX 2080Ti.

4.3. Comparison with Other Works

4.3.1. Performance on the BSDS500

We compare the proposed adaptive multi-level feature pyramid network via NMS Layer (AFPN_NMS) with some non-deep learning methods, including Canny, EGB (Efficient Graph-based) [28], Mshift (Mean Shift) [29], gPb-UCM [3], MCG (Multiscale Combinatorial Grouping) [30], SE, and OEF (Oriented Edge Forests) [31], as well as some recent deep-learning-based methods, including DeepContour [19], DeepEdge [18], HED [10], RCF [11], BDCN [20], etc. The default setting uses a VGG16 [11] backbone net, and we also compare RCF [32] using a ResNet backbone net.

The quantitative results are listed in Figure 4 and Table 1. Note that among all deep-learning-based methods, only our method has not been post-processed (using NMS) before evaluation. Figure 4 shows precision-recall curves of all methods, so it can be seen that the ODS F-measure of our method outperforms all of competing methods. The statistic comparison is shown in Table 1. It can be seen that AFPN_NMS ‡ outperforms HED, RCF ‡, BDCN ‡ by 2.7%, 1.7% and 0.9%, respectively in the ODS F-measure. When comparing with RCF-ResNet50 † and RCF-ResNet50 ‡, the ODS F-measures of AFPN_NMS-ResNet50 † and AFPN_NMS-ResNet50 ‡ are 2.2% and 2.5% higher than them, respectively.

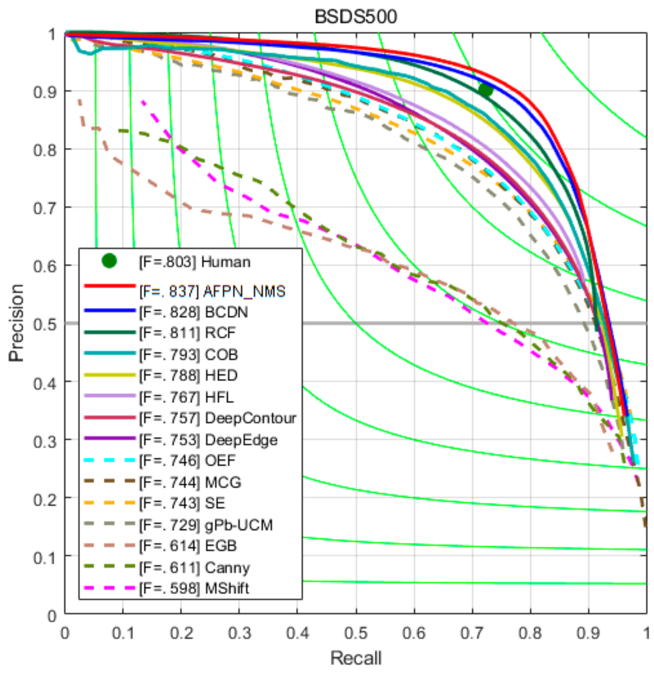

Figure 4. The evaluation results standard BSDS500 dataset.

In terms of algorithm speed, we mainly test HED, RCF, BDCN, and AFPN, since they all use vgg16 or ResNet50 as the backbone network. We test them on the same computing platform (GTX 2080Ti), so that a fair comparison can be made. It can be seen from Table 1 that AFPN_NMS runs at about 23 FPS (Frames Per Second) for edge detection, on par with BDCN. AFPN_NMS is slightly slower than HED and RCF because our method adds a non-maximum suppression layer, which consumes a small amount of calculation. However, the predicted results of HED, RCF, and BDCN are thick and need post-processing before the evaluation. Our network can directly output sharp boundaries without post-processing and achieve end-to-end training. AFPN_NMS with VGG16 achieves a good trade-off between effectiveness and efficiency, with the ODS F-measure of 0.837 and the speed of 6 FPS. Using ResNet50 as the backbone network, the ODS F-measure of AFPN_NMS increases from 0.837 to 0.839, though the speed drops from 6 FPS to 4 FPS.

The qualitative comparisons in Figure 5 suggest that, based on the non-maximum suppression layer, our method generates sharper boundaries. From a holistic perspective, the predicted results of BDCN and ours are closest to the ground truths. Compared with BDCN, it can be seen from the red dashed box that our method can better extract detailed features. This is due to the fact that PCM helps to extract multi-scale features. It can be seen from the blue dashed box that HED, RCF, and BDCN fail to suppress non-semantic edges. Our method can suppress non-semantic edges due to the use of the ODS F-measure loss function.

Table 1. Comparison with some competitors on the BSDS500 dataset. † indicates trained with additional PASCAL-Context data. ‡ indicates the fused result of multi-scale images. * means GPU time.

Method	ODS	OIS	FPS
Human [33]	0.803	0.803	-
Canny [7]	0.611	0.676	-
EGB [28]	0.614	0.658	-
Mshift [29]	0.598	0.645	-
gPb-UCM [3]	0.729	0.755	-
MCG [30]	0.744	0.777	-
SE [9]	0.743	0.763	-
OEF [31]	0.746	0.770	-
DeepContour [19]	0.757	0.776	-
DeepEdge [18]	0.753	0.772	-
HED [10]	0.788	0.808	34 *
RCF [11]	0.798	0.815	33 *
RCF † [11]	0.806	0.823	33 *
RCF ‡ [11]	0.811	0.830	9 *
RCF-ResNet50 † [32]	0.808	0.825	22 *
RCF-ResNet50 ‡ [32]	0.814	0.833	6 *
BDCN [20]	0.806	0.826	23 *
BDCN † [20]	0.820	0.838	23 *
BDCN ‡ [20]	0.828	0.844	6 *
AFPN_NMS	0.815	0.833	23 *
AFPN_NMS †	0.829	0.847	23 *
AFPN_NMS ‡	0.837	0.854	6 *
AFPN_NMS-ResNet50 †	0.830	0.849	15 *
AFPN_NMS-ResNet50 ‡	0.839	0.856	4 *

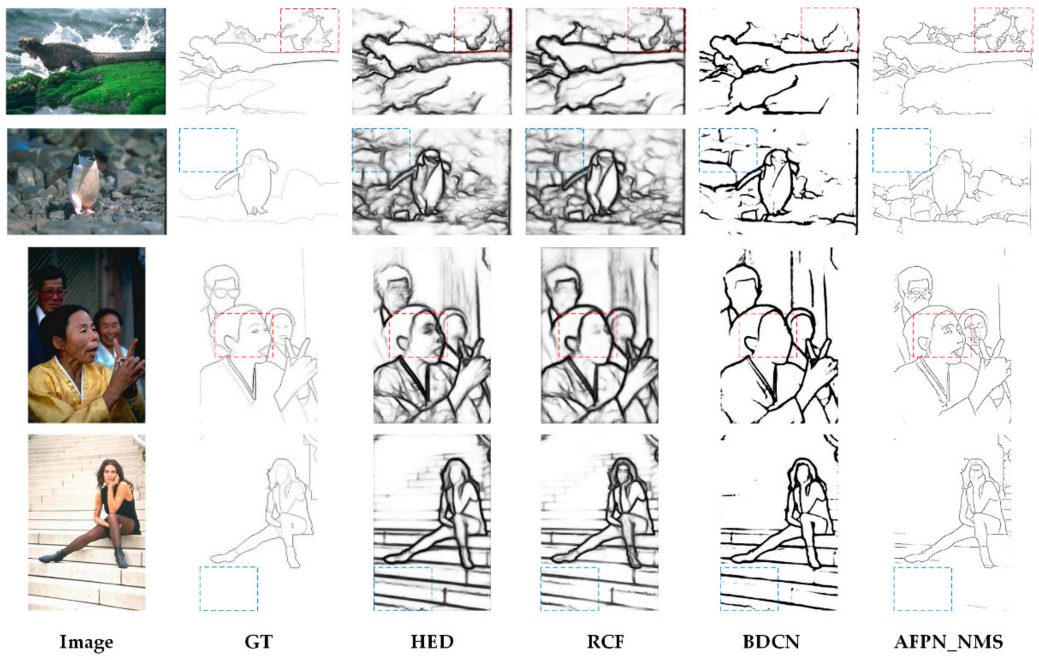

Figure 5. Comparison of edge detection results on the BSDS500 test dataset. Note that all the predictions are end-to-end outputs and not post-processed.

4.3.2. Performance on the NYUDv2

We compare our approach with some famous competitors. The evaluation results on the NYUDv2 are summarized in Figure 6 and Table 2, respectively.

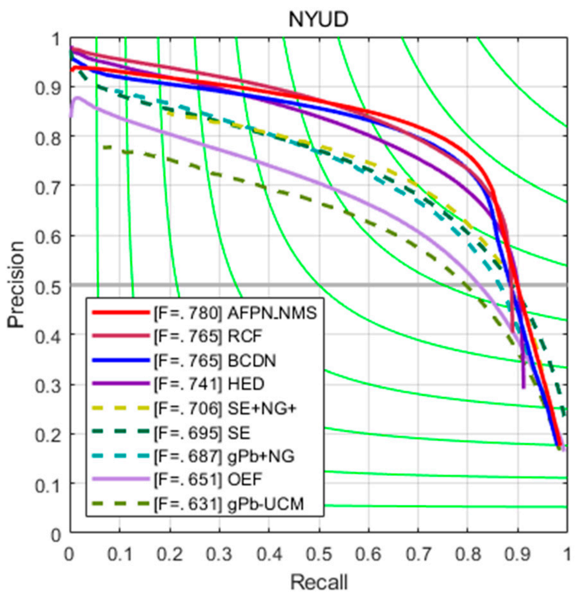

Figure 6. The evaluation results standard NYUDv2 dataset.

Table 2. Comparison with recent works on the NYUDv2. * means GPU time.

Method	ODS	OIS	FPS
OEF [31]	0.651	0.667	-
gPb-UCM [3]	0.631	0.661	-
SE [9]	0.695	0.708	-
HED-HHA [10]	0.681	0.695	23 *
HED-RGB [10]	0.717	0.732	23 *
HED-RGB-HHA [10]	0.741	0.757	11 *
RCF-HHA [11]	0.705	0.715	23 *
RCF-RGB [11]	0.729	0.742	23 *
RCF-RGB-HHA [11]	0.757	0.771	11 *
RCF-ResNet50-RGB-HHA [32]	0.781	0.793	8 *
BDCN-HHA [20]	0.707	0.719	15 *
BDCN-RGB [20]	00.748	0.763	15 *
BDCN-RGB-HHA [20]	0.765	0.781	7 *
AFPN_NMS-HHA	0.720	0.729	15 *
AFPN_NMS-RGB	0.756	0.772	15 *
AFPN_NMS-RGB-HHA	0.780	0.794	7 *
AFPN_NMS-ResNet50-RGB-HHA	0.790	0.802	5 *

The quantitative results are listed in Figure 6 and Table 2. Note that among all deep-learning-based methods, only our method has not been post-processed (using NMS) before evaluation. Figure 6 shows precision-recall curves of all methods; it can be seen that the ODS F-measure of our method outperforms all of the competing methods. The statistic comparison is shown in Table 2. In terms of HHA data, it can be seen that the proposed method outperforms HED, RCF, and BDCN by 3.9%, 1.5%, and 1.3%, respectively, in the ODS F-measure. In terms of RGB data, the proposed method outperforms HED, RCF, and BDCN by 3.9%, 2.7%, and 0.8%, respectively. On the merged HHA-RGB data, the proposed method outperforms HED, RCF, and BDCN by 3.9%, 2.3%, and 1.5%, respectively, in the ODS F-measure. For merging RGB-HHA data, AFPN_NMS-ResNet50 is 0.9% higher than RCF-ResNet50. AM-FPN with ResNet50 [26] improves a 1.0% ODS F-measure, when compared with AFPN_NMS with VGG16 [13]. ResNet50 can further improve the performance with more conv layers.

4.4. Ablation Study

In this section, in order to verify the effects of the components proposed in this paper, we conducted a detailed ablation experiment on the BSDS500 to compare the performance after using or removing these components. We train the network on the BSDS500 training set and evaluate it on the validation set.

Firstly, to prove the effectiveness of our proposed network structure, HED is chosen as the benchmark because the backbone of AFPN is the same as the main network of HED. To be fair, we use the same VGG16 pre-training parameters and the same training hyper parameters.

The quantitative results are listed in Table 3. AFPN achieve substantially higher results than HED. This proves the effectiveness of the bottom-up pyramid structure network. Compared with HED, it captures multi-scale spatial contexts and does not significantly increase network parameters. For the edge detection task, AFPN is 1.58% ODS F-measure higher than HED.

Table 3. Effect of the network structure.

Architecture	ODS	OIS
HED	0.7786	0.7993
AFPN	0.7944	0.8116

In order to prove the effectiveness of non-maximum suppression, we compared the two-network HED and the proposed AFPN, which are with and without the non-maximum suppression layer, respectively. When evaluating the results, post-processing is added to the output of the network that lacks a non-maximum suppression layer, and there is no need for the network that has a non-maximum suppression layer. The results are as follows:

It can be seen from Table 4 that after adding NMS, the ODS F-measure of HED is 0.39% higher than before. When adding NMS to AFPN, the ODS F-measure is 0.52% higher than before. It proves the effectiveness of a non-maximum suppression layer. The most important thing is that the network can directly output a sharp edge without post-processing.

Table 4. Effect of non-maximum suppression layer.

Architecture	ODS	OIS
HED	0.7786	0.7993
HED_NMS	0.7825	0.8035
AFPN	0.7944	0.8116
AFPN_NMS	0.7996	0.8174

The effect of the coefficient α in Equation (3) is tested experimentally. The experimental results are summarized in Table 5:

Table 5. Effect of coefficient α.

α	ODS	OIS
0.2	0.7882	0.7987
0.4	0.7996	0.8174
0.6	0.7978	0.8159
0.8	0.7953	0.8132

It can be seen from Table 5 that if α is set too small, such as 0.2, the non-maximum suppression layer will suppress some useful boundaries in training. The ODS F-measure score is the highest when α is set to 0.4. When α is set too high, such as 0.6 and 0.8, NMS does not have much effect on suppressing non-edge pixels.

Since the convergence of the network using only the ODS F-measure as the loss function is very slow, the cross-entropy loss and the ODS F-measure loss are combined to train the network. As shown in Figure 7, the annotation styles of the BSDS500 and the NYUDv2 are obviously different. The BSDS500 focuses on semantically meaningful contours, while the NYUDv2 has many detailed edges. The ODS F-measure loss is proposed to suppress non-semantic edges, so the value of γ in the BSDS500 is greater than that in the NYUDv2. The number of negative samples (N_{neg}) of the BSDS500 and the NYUDv2 are about 1.5×10^5 and 2.2×10^5, respectively. We manually select the value of ε to determine the value of γ. The effect of the balance factor ε of the binary cross-entropy and the ODS F-measure loss is tested experimentally. The experimental results are summarized in Table 6:

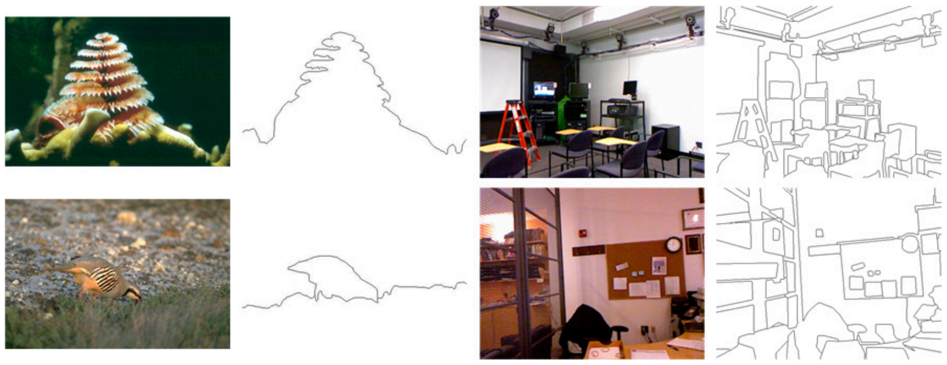
Figure 7. Comparison of annotation styles of the BSDS500 and the NYUDv2. The first two columns are origin images and ground truths of the BSDS500, respectively. The last two columns are origin images and ground truths of the NYUDv2, respectively.

Table 6. Effect of balance factor on the edge detection performance on the BSDS500.

ε	ODS	OIS
100	0.7956	0.8122
200	0.7976	0.8176
300	0.8032	0.8212
400	0.7966	0.8162
500	0.7946	0.8111

If ε is set to 200, the ODS F-measure loss function does not have much effect and the ODS F-measure is only 0.12% higher than without the ODS F-measure loss. The ODS F-measure score is the highest when ε is set to 300. The ODS F-measure is 0.88% higher than using only the binary cross-entropy. Both quantitative and qualitative results prove the effectiveness of the proposed ODS F-measure loss. If ε is set to 500, the ODS F-measure loss function has an adverse effect, and the edge suppression is too strong, which is in line with our original intention of designing the ODS F-measure loss.

Table 7 shows that the ODS F-measure score is the highest when ε is set to 2000. Experiments show that ε in the BSDS500 is less than that in the NYUDv2, and the corresponding γ in the BSDS500 is greater than that in the NYUDv2. This is because the ODS F-measure loss is proposed to suppress non-semantic edges, while the BSDS500 focuses more on semantical contours. These results also demonstrate the effectiveness of the ODS F-measure loss.

Table 7. Effect of balance factor on the edge detection performance on the NYUDv2.

ε	ODS	OIS
500	0.7382	0.7426
1000	0.7436	0.7493
2000	0.7501	0.7632
3000	0.7420	0.7463
4000	0.7270	0.7332

5. Conclusions

To better combine different levels of features, we propose an adaptive multi-level feature pyramid network (AFPN) for edge detection. Furthermore, a pyramid context module (PCM) is introduced which includes dilated convolution to achieve multi-scale feature. In order to obtain sharp edges without post-processing and achieve end-to-end training, we propose a non-maximum suppression layer and an optimal dataset scale (ODS) F-measure loss function. The experiments show that the proposed method achieves state-of-the-art results on the BSDS500 and the NYUDv2. In future work, we plan to extend the use of our network architecture to other tasks, such as salient object detection and semantic segmentation.

Author Contributions: G.S. and H.Y. designed the algorithms, performed the experiments, and analyzed the experimental data. X.J. and M.F. contributed to data analysis, checking, and correcting. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: A publicly available dataset was analyzed in this study. These data can be found here: https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html (accessed on 10 December 2021), https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html#bsds500 (accessed on 10 December 2021).

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

References

1. Ullman, S.; Basri, R. Recognition by linear combinations of models. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991**, *13*, 992–1006. [CrossRef]
2. Ferrari, V.; Fevrier, L.; Jurie, F.; Schmid, C. Groups of adjacent contour segments for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 36–51. [CrossRef] [PubMed]
3. Arbelaez, P.; Maire, M.; Fowlkes, C.; Malik, J. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 898–916. [CrossRef] [PubMed]
4. Malik, J.; Belongie, S.; Leung, T.; Shi, J. Contour and texture analysis for image segmentation. *IJCV* **2001**, *43*, 7–27. [CrossRef]
5. Cheng, M.-M.; Zhang, Z.; Lin, W.-Y.; Torr, P. BING: Binarized normed gradients for objectness estimation at 300 fps. *Comput. Vis. Media* **2019**, *5*, 3–20. [CrossRef]
6. Kittler, J. On the accuracy of the sobel edge detector. *Image Vis. Comput.* **1983**, *1*, 37–42. [CrossRef]
7. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *8*, 679–698. [CrossRef] [PubMed]
8. Lim, J.J.; Zitnick, C.L.; Dollár, P. Sketch tokens: A learned mid-level representation for contour and object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 25–27 June 2013.
9. Dollar, P.; Zitnick, L.C. Fast Edge Detection Using Structured Forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1558–1570. [CrossRef] [PubMed]
10. Xie, S.; Tu, Z. Holistically-nested edge detection. *Int. J. Comput. Vis.* **2015**, *125*, 3–18. [CrossRef]
11. Liu, Y.; Cheng, M.-M.; Hu, X.; Bian, J.-W.; Zhang, L.; Bai, X.; Tang, J. Richer convolutional features for edge detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3000–3009.
12. Deng, R.; Shen, C.; Liu, S.; Wang, H.; Liu, X. Learning to predict crisp boundaries. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
13. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014.
14. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
15. Wang, C.; Zhong, C. Adaptive Feature Pyramid Networks for Object Detection. *IEEE Access* **2021**, *9*, 107024–107032. [CrossRef]
16. Zhang, Z.; Xing, F.; Shi, X.; Yang, L. SemiContour: A Semi-supervised Learning Approach for Contour Detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
17. Ganin, Y.; Lempitsky, V. N4 fields: Neural network nearest neighbor fields for image trans-forms. In Proceedings of the Asian Conference on Computer Vision (ACCV), Singapore, 1–5 November 2014.
18. Bertasius, G.; Shi, J.; Torresani, L. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.

19. Shen, W.; Wang, X.; Wang, Y.; Bai, X.; Zhang, Z. Deep-Contour: A deep convolutional feature learned by positive-sharing loss for contour detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
20. He, J.; Zhang, S.; Yang, M.; Shan, Y.; Huang, T. BDCN: Bi-Directional Cascade Network for Perceptual Edge Detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
21. Chen, L.C.; Papandreou, G.; Schroff, F.; Adam, H. Rethinking atrous convolution for semantic image segmentation. *arXiv* **2017**, arXiv:1706.05587.
22. Rijsbergen, C.J.V. A non-classical logic for information retrieval. *Comput. J.* **1986**, *29*, 481–485. [CrossRef]
23. Abdou, I.; Pratt, W. Quantitative Design and Evaluation of Enhancement/Thresholding Edge Detectors. *Proc. IEEE* **1979**, *67*, 753–763. [CrossRef]
24. Silberman, N.; Hoiem, D.P.; Fergus, R. Indoor segmentation and support inference from RGBD images. In Proceedings of the European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012.
25. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; Devito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. In Proceedings of the NIPS 2017 Autodiff Workshop: The Future of Gradient-Based Machine Learning Software and Techniques, Long Beach, CA, USA, 9 December 2017.
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
27. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255.
28. Felzenszwalb, P.F.; Huttenlocher, D.P. Efficient graph-based image segmentation. *IJCV* **2004**, *59*, 167–181. [CrossRef]
29. Comaniciu, D.; Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal.* **2002**, *24*, 603–619. [CrossRef]
30. Arbeláez, P.; Pont-Tuset, J.; Barron, J.T.; Marques, F.; Malik, J. Multiscale combinatorial grouping. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 328–335.
31. Hallman, S.; Fowlkes, C.C. Oriented edge forests for boundary detection. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1732–1740.
32. Liu, Y.; Cheng, M.-M.; Hu, X.; Bian, J.-W.; Zhang, L.; Bai, X.; Tang, J. Richer convolutional features for edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 1939–1946. [CrossRef] [PubMed]
33. Mély, D.A.; Kim, M.; McGill, M.; Guo, Y.; Serre, T. A systematic comparison between visual cues for boundary detection. *Vis. Res.* **2016**, *120*, 93–107. [CrossRef] [PubMed]