Refined Edge Detection With Cascaded and High-Resolution Convolutional Network

Article *in* Pattern Recognition · June 2023

DOI: 10.1016/j.patcog.2023.109361

5 authors, including:

Omar Elharrouss
United Arab Emirates University
84 PUBLICATIONS **1,774** CITATIONS
SEE PROFILE

Youssef Hmamouche
Mohammed VI Polytechnic University
22 PUBLICATIONS **107** CITATIONS
SEE PROFILE

Assia Kamal Idrissi
Mohammed VI Polytechnic University
16 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Refined Edge Detection With Cascaded and High-Resolution Convolutional Network

Omar Elharrouss[a], Youssef Hmamouche[a], Assia Kamal Idrissi[a], Btissam El Khamlichi[a], Amal El Fallah-Seghrouchni[a]

[a]*International Artificial Intelligence Center of Morocco (Ai Movement) - University Mohammed VI Polytechnique*

Abstract

Edge detection is represented as one of the most challenging tasks in computer vision, due to the complexity of detecting the edges or boundaries in real-world images that contains objects of different types and scales like trees, building as well as various backgrounds. Edge detection is represented also as a key task for many computer vision applications. Using a set of backbones as well as attention modules, deep-learning-based methods improved the detection of edges compared with traditional methods like Sobel or Canny. However, images of complex scenes still represent a challenge for these methods. Also, the detected edges using the existing approaches suffer from non-refined results with erroneous edges. In this paper, we attempted to overcome these challenges for refined edge detection using a cascaded and high-resolution network named (CHRNet). By maintaining the high resolution of edges during the training process, and conserving the resolution of the edge image during the network stage, sub-blocks are connected at every stage with the output of the previous layer. Also, after each layer, we use

batch normalization layer with an active affine parameter as an erosion operation for the homogeneous region in the image. The proposed method is evaluated using the most challenging datasets including BSDS500, NYUD, and Multicue. The obtained results outperform the designed edge detection networks in terms of performance metrics and quality of output images.The code is available at: https://github.com/elharroussomar/chrnet/
Keywords: Edge detection, Convolutional neural networks, Deep learning, Scale-representation, Backbone.

1. Introduction

Extracting the salient edge from natural images represents a challenge for computer vision applications[1, 2]. Due to the complexity of images, the objects in them (trees, buildings, cars), and the collisions between the components of images, make the separation of the edges a difficult operation using statistical-based methods [3, 4, 5]. Nowadays, after the introduction of deep learning techniques as well as the development of the machine's performance with GPUs, this task becomes doable with convincing accuracy and the possibility of implementing it in real-time applications [7, 8].

Edge detection is the operation of extracting the contour of different objects and automatically ignoring the other details [9, 10]. It's also the operation of labeling the boundaries between the homogeneous parts in an image. This detection can be exploited by other computer vision tasks like object detection [11], image segmentation [12], and image reconstruction [13].

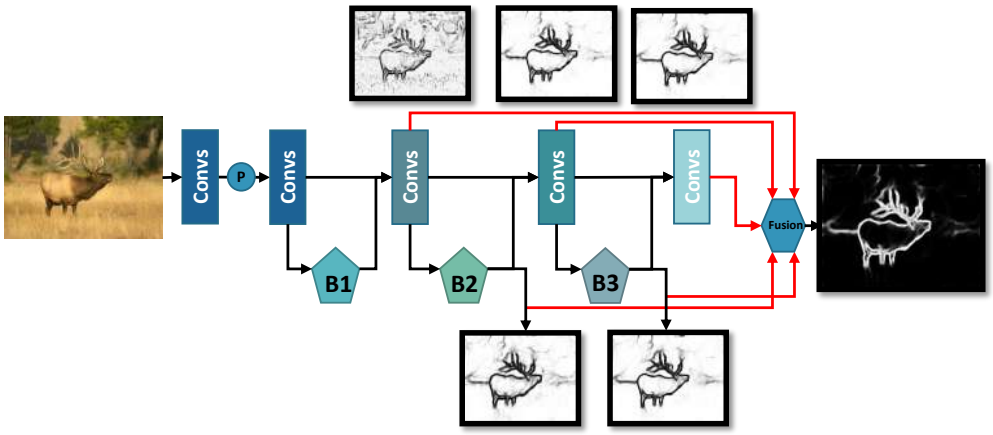


Figure 1: The proposed architecture for edge detection. *Convs* represents the list of convolutional layers. P is a pooling layer. B_i represents the block of convolutional and pooling layers that allows deep learning from the extracted features at each stage.

Also, besides RBG images the edge can be detected from infrared images [14]. This can be done with the analysis of low-level and high-level features that make the processing of the images cover all the components in it. These features are used also with deep learning models to reach a detection with high performance. Many methods have been proposed for this purpose, but the detection of object boundaries conserving some details about the objects affects the performance of these methods to achieve optimal results [15]. In addition, the scale variations, the shapes of objects, and the intensities of image regions affect the quality of detection for all the proposed methods.

In order to overcome the cited challenges, researchers developed different deep learning architectures [16, 17]. For example, some researchers exploited and adopted deep learning backbones like VGG, ResNet, DenseNet, and others, for constructing new edge detection models [18]. While some Con-

Table 1: Summarization of edge detection methods

Task	Method	Technique/Backbone	Dataset
Statistical-based	Martin et al. [1]	Brightness and Color Gradients, Classifier	Berkeley
	Arbelaez et al. [2]	Brightness, Color, and Texture Gradients, r	BSDS500
	Dollar et al. [3]	Structured random forests, boosted classifier	BSDS500, NYUD
	Mairal et al. [6]	Least-squares reconstruction errors minimization	PASCAL VOC
	Ren et al. [7]	Local sparse coding, SVM	BSDS500, NYUD
	Lim et al. [8]	sketch tokens filter, HOG	BSDS500, PASCAL VOC
	Romani et al. [9]	Variably Scaled Kernels (VSKs) interpolation	Simple images
	Mittal et al. [10]	Simulated triple thresholds	Simple images
	Wang et al. [14]	Spiking neural network	Infrared images
	Sert et al. [15]	Maximum norm entropy (EDA-NMNE)	Simple Images
Deep-learning-based	CAFENet [29]	Encoder-decoder/ ResNet-34	FSE-1000 and SBD-5
	HED [27]	Holistically-Nested Network /VGG16	BSDS500, NYUD
	LPCB [28]	ResNeXt blocks / VGG16	BSDS500, NYUD
	RCF [30]	Richer convolutional features (RCF)/ VGG16	BSDS500, NYUD, Multicue
	BDCN [32]	VGG16, Cascade network	BSDS500, NYUD, Multicue
	REDN [34]	DenseNet, Encoder-decoder	BSDS500, NYUD
	PiDiNet [35]	CNN, Dilation and spatial attention module	BSDS500, NYUD, Multicue
	RHN [36]	CNN, Residual VGG-16	BSDS500, NYUD, Multicue
	Li et al. [37]	ANDD matrices, CESM, CEDM	BSDS500

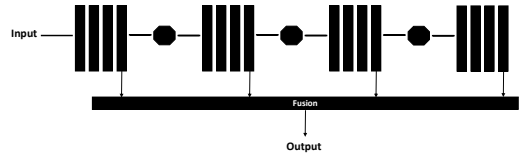
volutional Neural Networks (CNN) with a hierarchical representation are proposed to take benefit from different features generated by each block of layers [19]. These edge detection methods attempted to visualize the outputs of different layers [20]. While the development of edge predictions in different intermediate layers can be shown, which gives the researchers the possibility to enhance or change a part of the network to improve the quality of predicted edges [21]. Also, to enforce the first layers of the network to predict the edge from different scales, the last layers are chosen to work on a certain scale and enhance the output of the previous layers.

These methods used all convolutional features that represent multi-scale representations as used for computer vision tasks, then a combination of each stage output is performed [22]. While some of these methods used unified networks with encoder-decoder architectures [23]. Training these models, which

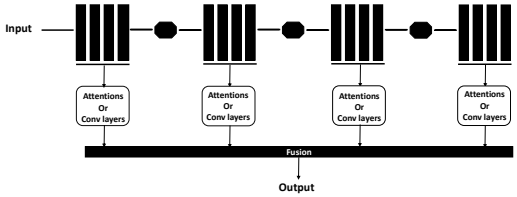
can contain millions of parameters, is costly in terms of time and memory space, especially when the networks are composed of attention modules. This also represents a challenge for these methods in the case of using the edge detector algorithm in real time.

Exploiting different intermediate layers outputs as well as minimizing the number of parameters in the networks, we proposed a new edge detection network with a refined representation. The proposed architecture consists of using different block outputs and avoiding the loss of certain features during pooling operations. This is performed by concatenating each block's results with the outputs of the previous layer as illustrated in Figure 1. The advantage of the proposed model is that the network is not composed of any dilation layers as well as without attention modules. In order to solve low-quality detection (non-maximum pixels and the noise around the edge region), we used batch-normalization with learnable affine parameters which makes the proposed network able to remove these kinds of noise around the edge region. After the experiments on many datasets including BSDS500, NYUD, and Multicue as well as compared with the state-of-the-art methods, the obtained results using the proposed architecture are more accurate in terms of quality, precision, and also outperform all methods, especially for NYUD dataset. In terms of training and testing time, the proposed model is less computational time and a shown difference in FPS values compared with the most performed methods.

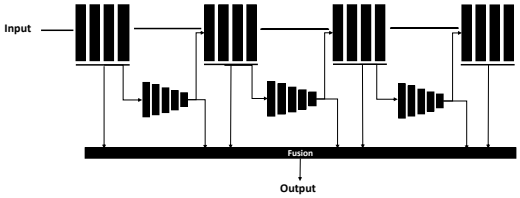
The paper sections are organized as follows. The related works are pre-



(a) Multi-scale representation



(b) Multi-scale representation with attention modules



(c) Our multi-scale representation

Figure 2: The structure of edge detection architectures. First architecture is used by HED [27]. Second architecture is exploited by RCF [30], BDCN [32], PiDiNet[35], and RHN [36] methods

sented in section 2. Section 3 describes the proposed edge detection method presented in section 4. The obtained results and discussion of them are provided in Section 5. A conclusion is presented in section 6.

2. Related works

Edge detection also named boundary or contour detection still one of the challenging tasks in computer vision using statistical or deep-learning-based methods. Statistical-based methods exploiting color, local brightness features [24], clustering algorithms [25], or local image patches [26] for detect-

ing the edge in an image. The statistical-based methods can work on a set of data while the detection is based on the selected features which makes these methods accurate for some types of scenarios and less performing for others [4, 7, 6, 8]. Mainly the low-level features like the pixel intensities and color gradients [9, 10], the object texture is exploited with learning approaches to detect the content boundaries in the images [14, 15]. Even so, these methods have limitations such as low real-time accuracy, high-level information extraction, and sensitivity to scale and environmental changes.

For deep-learning-based methods that used Convolutional neural networks trained on large-scale datasets [29, 30, 31], the possibility to analyze a large set of scenarios and type of scene becomes possible which makes such a method to be generic. The proposed method attempted to work on scale representations to extract the edge at each block of the network. Then these features are fused to construct the final edge detection results. For that, some authors proposed a unified network based on the existing feature extraction backbone. For example, in [27] VGG-16 network is used for holistically-Nested edge detection architecture (HED). The same backbone has been used by Deng et al. [28] for boundaries detection (LPCB). LPCB method is a unified method (no multi-scale representation) that consists of connected parallel layers with ResNeXt blocks. Instead of predicting the edge directly from the input image, the authors in [29] start first by segmenting the salient object before extracting the edge from the segmented results using a deep learning architecture named CAFENet.

For example, in [30] the authors proposed an architecture named richer convolutional features (RCF) that exploited the VGG-16 backbone of 13 layers divided into 5 stages, which gives the ability to extract the features from multilevel and multi-scale representations. In the same context, the authors in [31] proposed a Bi-Directional Cascade Network (BDCN) architecture that consists of using each block of the CNN networks with a Scale Enhancement Module (SEM) for generating multi-scale features, then concatenating all outputs to obtain the final edge map.

In the same context, the authors in [32] proposed a Cascaded Network for edge detection named BDCN. The proposed network consists of a set of blocks separated by pooling layers and used VGG16 as a backbone. Each block output is taken into consideration for the final edge results. The scale variation is handled using a Scale Enhancement Module (SEM) connected to each convolutional layer in the network. The SEM Module allows multi-scale representations for the edge detection learning process. Another method named REDN is proposed in [34]. Unlike the other methods that used unified networks with a succession of blocks of convolutional and pooling layers, REDN is an encoder-decoder method for edge detection. The proposed architecture exploits DenseNet architecture as the encoder of the model, while the transposed convolutions are used for the decoder side.

Like in [31, 32], the authors in [35] proposed a deep learning architecture for edge detection named PiDiNet. The proposed method exploits the feature extraction network as the backbone of 4 blocks separated by pooling

layers. The dilation convolutions module and a spatial attention module are applied to the results of each block (stage). The results of all stages are used to generate the edge detection map. The proposed method has many versions including baseline, Tiny and small. In the same context, the authors in [36] proposed an edge detection model named RHN based on an extended version of VGG-16 named residual VGG-16 for feature extraction. The authors use each backbone stage, separated by pooling layers, outputs to be the input of a new block of convolutional and upsampling layers. All the outputs are concatenated to generate the final edge map. the authors in [37] proposed a deep learning method for edge detection, while the images are prepossessed before being introduced into the model. The method starts by extracting the features from R, G, and B images to compute anisotropic directional derivative (ANDD) matrices. the feature results are used for post-possessing with image decomposition by color edge strength maps (CESMs) technique and color edge direction maps (CEDMs). CESMs features are used for the classification stage to generate the edge map. From the experiment and the comparisons with deep learning methods, the proposed method is less accurate regarding the refinement of the edge regions. in the same context, the authors in [38] proposed a Lightweight Dense Convolutional (LDC) architecture whit a reduced number of parameters.

With the introduction of attention modules and their performance for Neural Language Processing (NLP) tasks, computer vision researchers attempted to combine self-attention with CNN-based architecture to improve

the performance of image processing tasks [39]. This led to the wide use of attention instead of convolutions networks for detection or segmentation purposes. In addition, it become the base of creating another technique based on attention named Transformers which is worked well in NLP [40]. Due to the successes of transformer networks in Neural Language Processing (NLP), the researchers attempted to exploit it on images for segmentation and detection tasks [41]. For that, the authors in [42] proposed a vision-transformer-based model for edge detection. The proposed method named EDTER consists of learning from transformer features, then detecting the edge after refining it to obtain the final results. Using the two-stage model, EDTER is costly in terms of complexity and number of parameters compared to the other models while it costs more than 900.0 GFLOPs in the two stages.

From all the obtained results of the proposed methods, we can observe that the architectures that are based on multi-scale presentations are the most accurate, also these methods used the same backbone for features extraction which is VGG-16. For the datasets used for training and evaluation of the proposed methods, the NYUD dataset represents the challenging dataset compared with the others like BSDS500 and Multicue datasets.

3. Proposed method

3.1. Formulation

Suppose that a sample of training set T denoted by (X, Y), While $X = \{x_j; j = 1; ...; |X|\}$ represent the raw of input image, and $Y = \{y_j; j =$

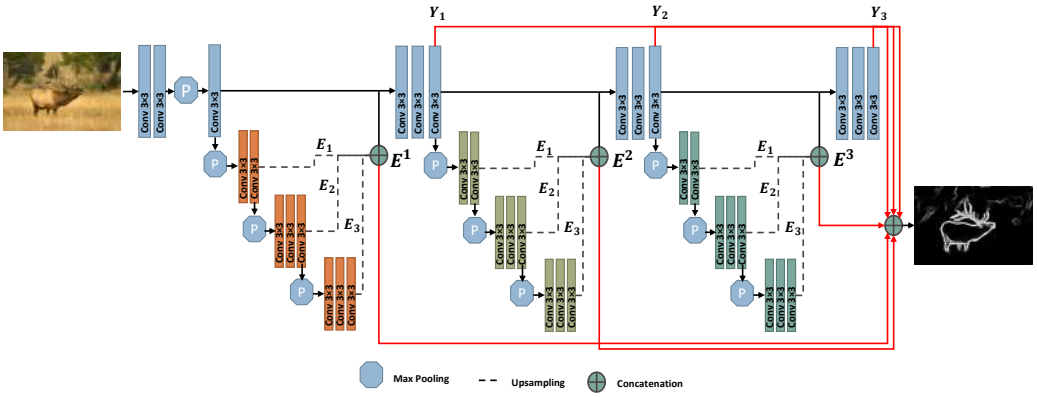


Figure 3: Flowchart of the proposed edge detection network.

$1; ...; |X|\}; y_j \in \{0; 1\}$ is ground-truth of edge map. Also Let $E = \{E_i^j; i = 1; ...; M; j = 1; ...; N\}$, while M represent the number of blocks in the network, and N represent the number of output of each block.

The edge of an object can be extracted during the learning process by removing the meaningless information as well as representing different scales. For that, we select the output of each block also the output of each scale of the network to get the final edge map. The edge map Y is a combination of the output of the S binary image of different scales of the network and the output E of each block that is used for conserving the resolution of the edge map. The two components of the proposed methodology can be expressed as follows.

$$Y = \sum_{s=1}^{S}(Y_s + \sum_{i=1}^{N}(E_i^s)) \tag{1}$$

where Y_s contains generated edge corresponding to a scale s, while the

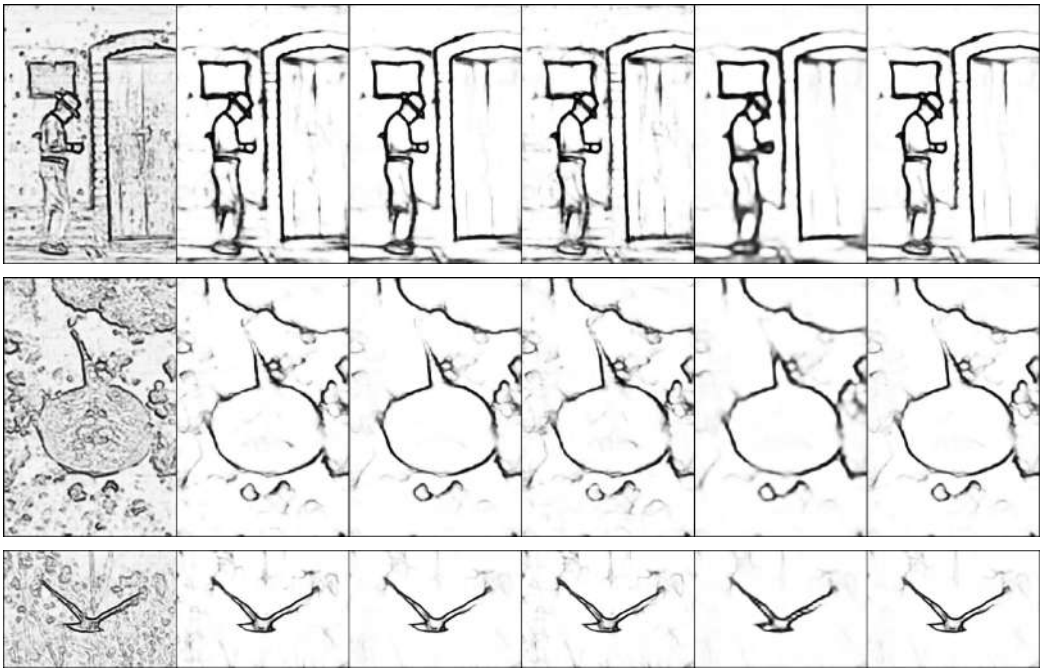


Figure 4: Feature maps generated by pixel at each stage of the proposed network.While the fire column represent the feature maps of the third convolutions layer before the first block. The column 2, 3, 4n and 5 represents the results of in Y^2, E^2, Y^3, E^3 respectively.

scale represents the part of the network that is composed of a set of convolutional layers without pooling operation, unlike the existing method.

Our purpose is to generate the edge without losing the features that represent the edges in the image with high resolution. The resolution conservation is performed by using only convolutional layers with combining them with the blocks at each time. Each block is the operation of scaling which is a set of convolutional and pooling layers that make adjacent convolutional layers depict image patterns at different scales.

During the training of an image X, the generated feature map of s-th convolutional layer $N_s(X)$ represents the generated edge Y_s. Y_s is used as

input of the Block B_s. After consecutive convolutional and pooling layers within the B_s, we generated M features $E_{i,i=1...M}$ that represent the edge maps of each scale (separated by pooling layer). the output of each scale is the combination of $E_{i,i=1...M}$ within the block B_s and the generated Y_s edge map. this operation ensures the resolution conservation that can be affected by using pooling operation. The loss of information generated edge map within s-th scale is less than the loss of Y_s with the same scale:

$$L_s(Y_s + \sum_{i=1}^{N}(E_i^s)) < L_s(Y_s)) \tag{2}$$

Using the concatenation of different scale output, the final estimated edge map, if we are suing 3 scale, can be defined as follows:

$$\hat{Y} = Y_1 + \sum_{i=1}^{N}(E_i^1) + Y_2 + \sum_{i=1}^{N}(E_i^2) + Y_3 + \sum_{i=1}^{N}(E_i^3) \tag{3}$$

The number of scale can very from 3 to ns, and the performance is improved proportionally to the number of scale ns. while the final results can be close to the ground-truth when the ns is high:

$$\hat{Y} = Y_1 + \sum_{i=1}^{N}(E_i^1) + Y_2 + \sum_{i=1}^{N}(E_i^2) + Y_3 + \sum_{i=1}^{N}(E_i^3) \tag{4}$$

Figure 3 represents the flowchart of the proposed architecture. While figure 4 illustrates all the outputs within each block of the network.

3.2. Proposed Architecture

The existing edge detection methods exploited different feature extraction networks as the backbone for their edge detection architectures. VGG-16 is the most used backbone for edge detection. These methods follow a common strategy which is the collaboration of different stages of the networks to elaborate the final edge results. This collaboration can take various forms as illustrated in Figure 2. The output of the last layers produces low-quality images as well as some regions are miss detected and contain noises, due to the use of a set of pooling layers of the unified network. Which can affect the quality of the edge map.

In order to maintain the high resolution of edges during the training process, we inspire by [33] as well as from the existing edge detection models like [30, 32, 34] using the multiple stages with the same scenario. The same strategy is followed in our proposed models for edge detection. The loss of information during the pooling operations makes the resolution of the final layer output very low and the contextual information can not be effective to be used during the fusion process. For that, and in order to conserve the resolution of the edge image during the network stage, we proposed a network that makes the pooling output at each stage should be connected and combined with the previous layer's output. As presented in Figure 3, the multi-resolution sub-network is connected with the output of the last layer which is connected always with the output of the convolution layers without using the pooling operation. The fusion operation is made at each level of

Table 2: Comparison of edge detection methods On BSDS500 dataset.

Method	**ODS**	**OIS**	**AP**	**FPS**
HED [27]	0.788	0.808	0.840	78
LPCB[28]	0.808	0.824	-	30
RCF [30]	0.806	0.823	-	30
RCF-MS [30]	0.811	0.830	-	8
REDN [34]	0.808	0.828	-	-
REDN (+PASCAL)[34]	0.761	0.785		-
PiDiNet-MS [35]	0.807	0.823	-	92
PiDiNet-small [35]	0.798	0.814		**148**
RHN [36]	0.817	0.833	-	33
CED [19]	0.794	0.811	-	-
CED [19]	0.815	0.833	-	-
Li et al. [37]	0.731	0.760	0.605	-
BDCN [32]	0.806	0.826	0.847	-
BDCN-MS[32]	0.828	0.844	**0.890**	-
EDTER [42]	0.824	0.841	0.880	-
Ours-3	0.787	0.788	0.801	86
Ours-MS-3	0.816	0.845	0.846	86
Ours-MS-5	**0.830**	**0.853**	0.870	32

the network while the final output is a concatenation of each stage of the network.

The number of blocks can vary from 3 to N. In the experiments, we used one 3 blocks and 5 blocks. we used in the last two blocks a $bacth - normalization$ layer after each convolution layer to enhance the quality of the generated edge by removing the non-maximum region between the edge. This is performed by activating the affine transformation parameter $Affine = True$ in the $batch-normalization$ function, which makes the $means-weight$ and $bias$ used in the learning process.

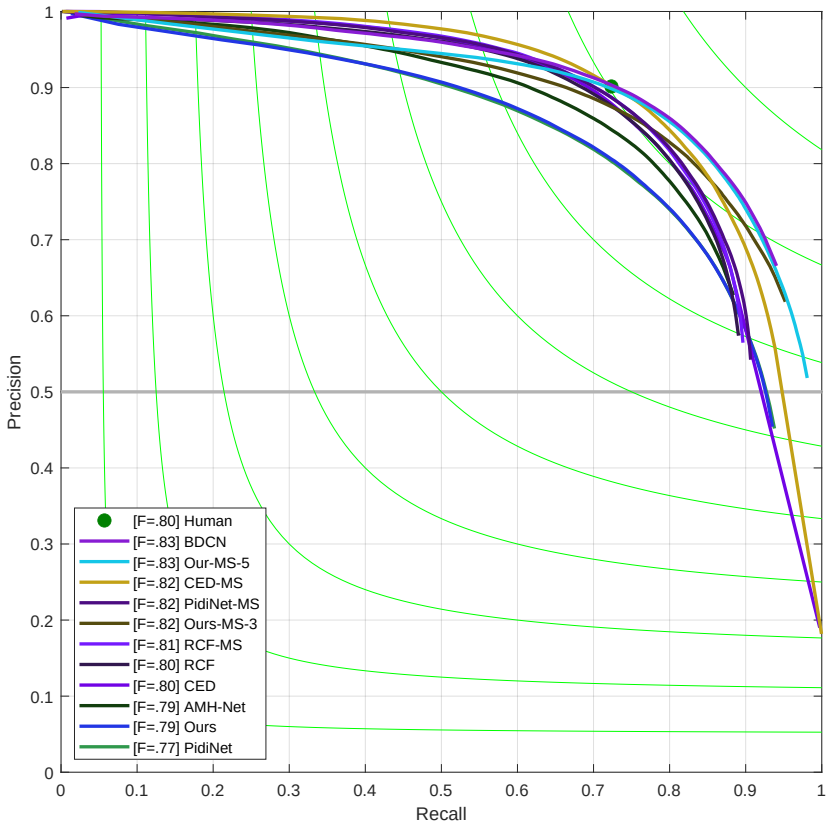


Figure 5: Precision-Recall curves of the proposed models compared with the best state-of-the-art methods on BSDS500 dataset.

4. Experimental results

This section provides a demonstration of the experimental results provided by the proposed method on three edge detection public datasets including BDSD500, NYUD, and Multicue. The evaluations and the comparisons have been performed to prove the effectiveness of the proposed architecture regarding the quality/quantity of the detection as well as the computational time for our method against the existing ones. The obtained results are com-

pared with a set of accurate state-of-the-art methods including HED [27], LPCB[28], REDN [34], RCF [30], PiDiNet [35], RHN [36],BDCN [32], and Li et al. [37]. The results are also presented by visualizing some examples for each dataset.

4.1. Implementation details

In order to train and evaluate the proposed method, the original split of each dataset has been used. Also, the data augmentation used by the existing method has been respected. For example, the training set of BSDS500 dataset was augmented with flipping (2×), scaling (3×), and rotation (16×), which lead to a dataset 96× larger than the original version.

We used for training the proposed model a laptop with 16 GB RAM, NVidia GPU 1070. The code is implemented with python and we used PyTorch library. The parameters of the model are the same as PidiNet method [35], while Adam optimizer is used and the learning rate of 0.005. The weight decay is set at 0.1. We used the same loss function of PidiNet method [35] with the parameters λ and η. While λ is set to 1.1 for both BSDS500 and Multicue, and 1.3 for NYUD. And the threshold η is set to 0.3 for both BSDS500 and Multicue. No η is needed for NYUD since the images are singly annotated.

The proposed method used parameters that can be adaptively related to the dataset used in training such as the number of blocks, and the number of epochs used for training the model. The number of blocks, used for training

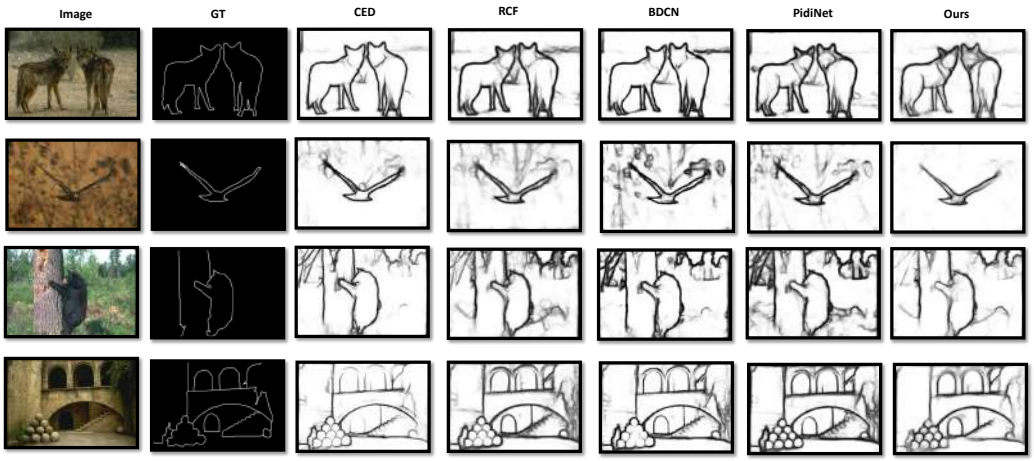


Figure 6: Some results of the detected edges on BSDS500 dataset.

the proposed model, is from 3 to 5 like in BDCN [32].

4.2. BSDS500 Dataset evaluation

BSDS500 dataset is composed of 500 images while 200 images are dedicated to the training set, 100 for validation, and 200 for testing. It's one of the most used datasets for detecting and extracting the edges in an image. Like the existing methods including RCF, PidiNet, and BDCN, for training the proposed method we used the same data augmentation which used the flipping, scaling, and rotation operations of 2×, 3×, and 16× respectively. The data also has been mixed with PASCAL-VOC data which is also augmented with the flipping operation and make it 2× larger. In the evaluation stage, we exploit the Non-maximum suppression (NMS) techniques to thin and normalize the detected edges. We compared the obtained results with the most accurate methods in the literature including HED [27],CED [19],

RCF [30], PidiNet [35], BDCN [32], and RHN [36]. The comparison is demonstrated using ODS, OIS, AP, and FPS metrics represented in Tables 2 with both single-scale and multiscale (MS) (the final results is the combination of different scale outputs) versions of the proposed method, Precision-Recall curves presented in Figure 5, and visualization of some obtained edge maps illustrated in Figure 6.

From Table 2, Figure 5, and Figure 6 we can observe the performance of the proposed methods compared with the existing methods. While from the table, the proposed method achieved 0.830 for ODS metrics, 0.853 for OIS metric and 0.870 for AP metric outperforming CED, RHN, and BDCN-MS methods that come in the second and third places by a difference for ODS metric of 15%, 13%, and 2% respectively. The same observation for the other methods as well as the output results presented with Precision-Recall curves in Figure 5, while the proposed method curve is more stable.

For the edge maps illustrated in Figure 6, the obtained results using the proposed method are more refined and with a high resolution of the detected edges compared with RCF, BDCN, and PidiNet methods. While CED results are improved and close to the obtained results. For example, the second result shows an accurate detection compared with the ground-truth image even though the image contains regions that can be classified as edges. The high resolution of the obtained results is the benefit of using the proposed refined network illustrated before with batch normalization.

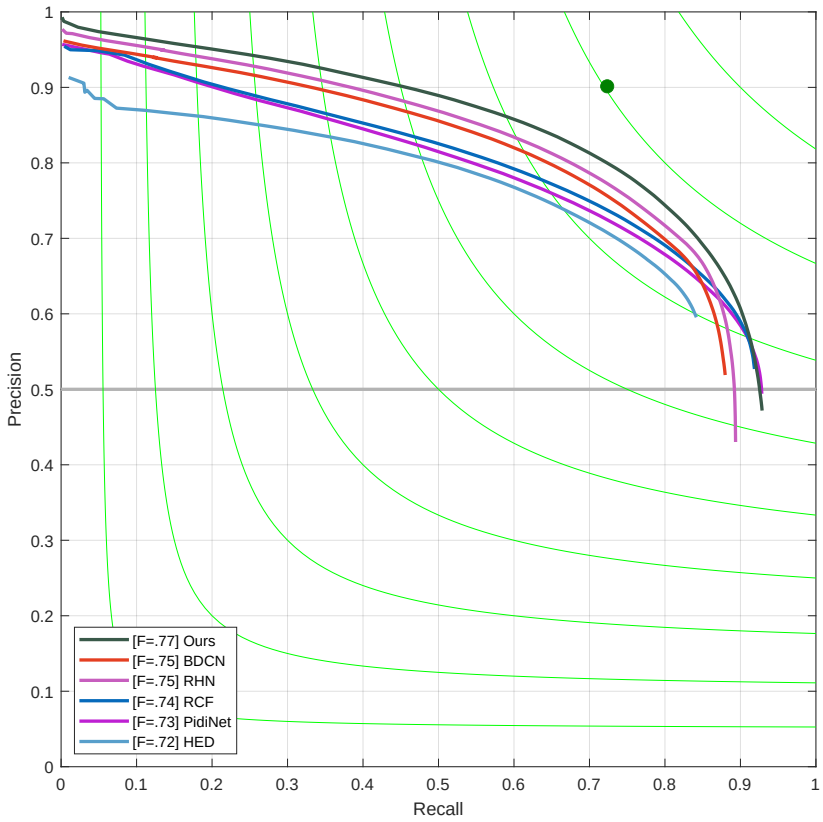


Figure 7: Precision-Recall curves of the proposed models compared with the best state-of-the-art methods on NYUD dataset.

4.3. NYUD Dataset evaluation

NYUDv2 is another edge detection dataset that contains two type sets of 1449 images, one of RGB images and the same image with depth presentation. The same processes of data augmentation have been performed with three operations including rotating (4×), flipping (2×), and scaling (3×). The proposed method has been trained on the RGB set, depth set (HHA), and the combination of the two sets (RGB+HHA). The same metrics exploited on BSDS500 have been used for evaluating the proposed method as

Table 3: Comparison of edge detection methods On NYUD dataset

Method	**RGB**		**HHA**		**RGB-HHA**		**FPS**
	ODS	OIS	ODS	OIS	ODS	OIS	
HED [27]	0.720	0.734	0.682	0.695	0.746	0.761	**62**
LPCB [28] (2018)	0.739	0.754	0.707	0.719	0.762	0.778	-
RCF [30] (2019)	0.743	0.757	0.703	0.717	-	-	20
BDCN [32] (2019)	0.748	0.763	0.707	0.719	0.765	0.781	
PiDiNet [35] (2021)	0.733	0.747	0.715	0.728	0.756	0.773	**62**
RHN [36] (2021)	0.751	0.762	0.711	0.721	0.772	0.789	24
Ours-3	0.729	0.745	0.718	0.731	0.750	0.774	51
Ours-MS-3	0.756	0.769	0.743	0.755	0.776	0.778	56
Ours-MS-5	**0.774**	**0.795**	**0.771**	**0.793**	**0.791**	**0.805**	34

Table 4: Comparison of edge detection methods On Multicue dataset

Method	**Boundary**		**Edge**		**FPS**
	ODS	OIS	ODS	OIS	
RCF [30]	-	-	0.857	0.862	15
BDCN [32]	0.836	0.846	0.891	0.898	9
PiDiNet [35]	0.818	0.830	0.855	0.860	17
RHN [36]	0.841	0.856	0.896	0.905	-
Ours-Ms-5	**0.859**	**0.863**	**0.907**	**0.922**	**23**

well as the state-of-the-art methods such as ODS and OIS.

For evaluating the proposed method, we used a maximum tolerance of 0.011 instead of 0.0075 used in BSDS500 dataset, because the size of NYUD images is larger than BSDS500 dataset ones. We Also performed the training, testing, and evaluation of the proposed model on the three parts of the dataset including, the RGB set, HHA set, and RGB+HHA set. Table 3 represents the obtained results using single and multi-scale (MS) models compared with state-of-the-art methods for the three parts. While Figure 7 illustrates the Precision-Recall curves of these methods on RGB set. From

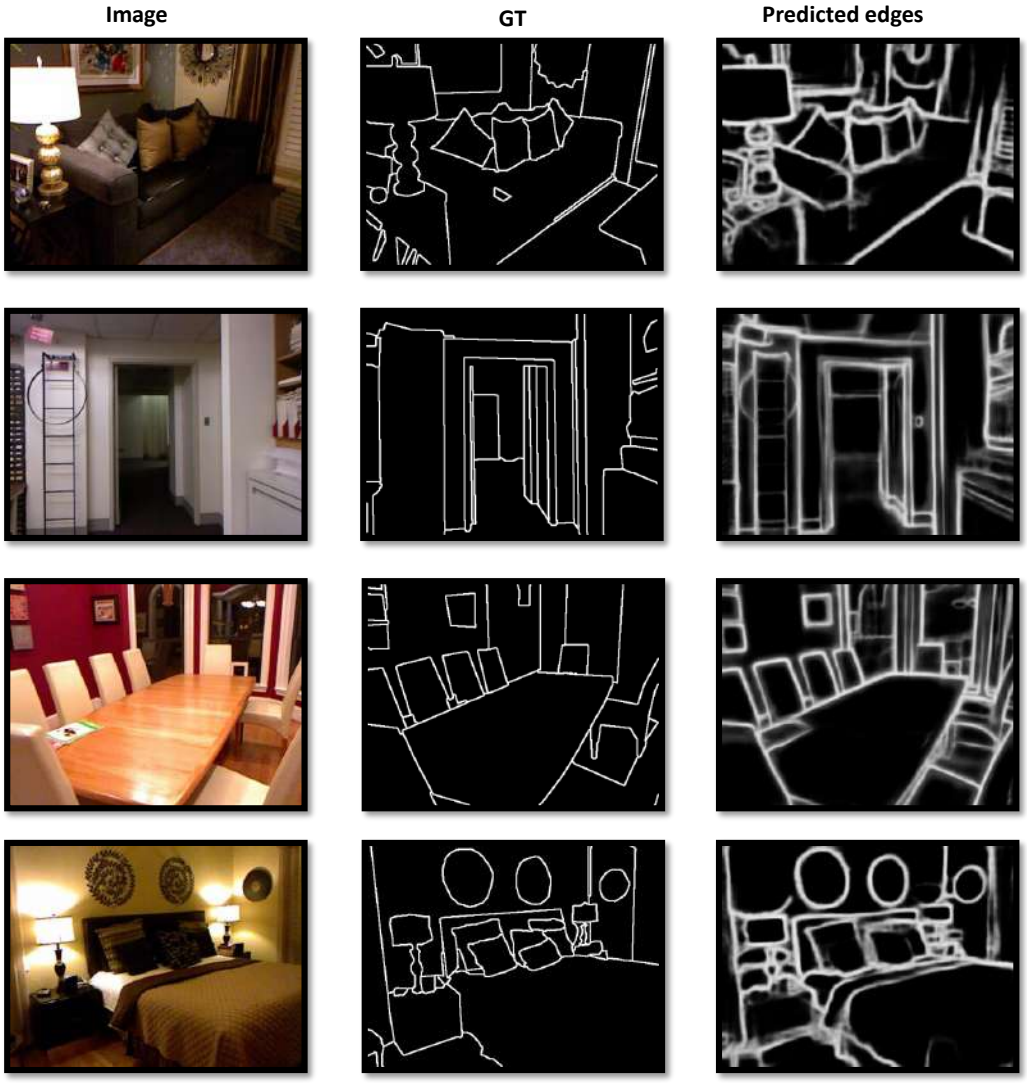


Figure 8: Some results of the detected edges from NYUD dataset.

Table 3 we can see that the proposed method outperforms the other methods for the two metrics including ODS and OIS. While the proposed method with MS-3 and MS-5 achieved 0.756 and 0.774 for the ODS metric on the RGB set and outperform BDCN with 2.8%, PidiNet with 4.3%, and RHN with 2.5%.

This observation is true also on the HHA set and RGB+HHA set. Figure 7 proves the obtained results in Table 3, while we can see that the Precision-Recall curve generated by the proposed method is more stable compared to the state-of-the-art methods.

In addition to the quantitative results represented in Table 3, Figure 8 illustrates some obtained results using the proposed method on the NYUD dataset. From the visualized results we can see the quality of the detected edges as well as the ability of the proposed method to detect edge regions without 'ghosts' that represent the unsuspected edges in some regions that do not contain any edge.

4.4. Multicue Dataset evaluation

Another dataset for edge detection named Multicue contains 100 real-world images of natural scenes and is represented as a challenging edge detection dataset. Multicue is labeled with edges and boundaries. The dataset has been augmented with flipping (2×), scaling (3×), and rotation (16×) operations. The proposed method has been evaluated using ODS and OIS metrics. Table 4 represents the obtained results using each method for edge and boundary labels. From the table, we can see that the proposed method and RHN method reached the best ODS and OIS values for edge and boundary representations. while the proposed method is better than RHN by 1% for ODS and OIS on boundary label and 1% for ODS and 2% for OIS on edge label.

5. Conclusion

This paper proposes a refined edge detection Method for detecting the contour boundaries of image content. The proposed method consists of detecting edges benefiting from the multi-scale representation of the network as well as conserving the high resolution of the output map. Unlike the existing methods, this is ensured by using the interconnection between consecutive parts of the network, exploiting the refined batch normalization layer, and involving the output of the first layers in the final results using fusion operation. After that, we evaluated the proposed method on three competitive datasets. The obtained results are compared with the best edge detection methods. The obtained results and the achieved promising performance rates in the different scenarios. Using the proposed method with 3 blocks (Ours-3 and Ours-MS-3), 86 frame per second (FPS) can be processed and with the obtained performance accuracy the use of the proposed method is suitable for real-time it useful for semantic edge segmentation as a perspective for the next work plan. Since with a basic camera, we can record at 60 FPS in Full HD quality.

References

[1] S. Guiming, S. Jidong, Multi-scale harris corner detection algorithm based on Canny edge-detection, in: 2018 IEEE International Conference on Computer and Communication Engineering Technology, 2018, pp. 305–309.

[2] Hallman, S., and Fowlkes, C. C. (2015). Oriented edge forests for boundary detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1732-1740).

[3] Lopez-Molina, C., Galar, M., Bustince, H., and De Baets, B. (2014). On the impact of anisotropic diffusion on edge detection. Pattern Recognition, 47(1), 270-281.

[4] S. Zheng, Z. Tu, and A. Yuille. Detecting object boundaries using low-,mid-, and high-level information. In CVPR, 2007.

[5] P.A. Flores Vidal, G. Villarino, D. Gómez, J. Montero, A new edge detection method based on global evaluation using supervised classification algorithms, Int. J. Comput. Intell. Syst. 12 (2019) 367–378.

[6] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In ECCV, 2008

[7] X. Ren and B. Liefeng. Discriminatively trained sparse code gradients for contour detection. In NIPS, 2012.

[8] J. Lim, C. L. Zitnick, and P. Dollar. Sketch tokens: A learned mid-level representation for contour and object detection. In CVPR, 2013.

[9] Romani, L., Rossini, M., & Schenone, D. (2019). Edge detection methods based on RBF interpolation. Journal of Computational and Applied Mathematics, 349, 532-547.

[10] Mittal, M., Verma, A., Kaur, I., Kaur, B., Sharma, M., Goyal, L. M., ... & Kim, T. H. (2019). An efficient edge detection approach to provide better edge connectivity for image analysis. IEEE access, 7, 33240-33255.

[11] Ji, G. P., Zhu, L., Zhuge, M., & Fu, K. (2022). Fast camouflaged object detection via edge-based reversible re-calibration network. Pattern Recognition, 123, 108414.

[12] Xu, X., Chen, J., Zhang, H., Han, G. (2022). SA-DPNet: Structure-aware dual pyramid network for salient object detection. Pattern Recognition, 127, 108624.

[13] Lin, J., Cai, Y., Hu, X., Wang, H., Yuan, X., Zhang, Y., ... Van Gool, L. (2022). Coarse-to-fine sparse transformer for hyperspectral image reconstruction. arXiv preprint arXiv:2203.04845.

[14] Wang, B., Chen, L. L., & Zhang, Z. Y. (2019). A novel method on the edge detection of infrared image. Optik, 180, 610-614.

[15] Eser, S. E. R. T., & Derya, A. V. C. I. (2019). A new edge detection approach via neutrosophy based on maximum norm entropy. Expert Systems with Applications, 115, 499-511.

[16] Le, M., & Kayal, S. (2021, July). Revisiting Edge Detection in Convolutional Neural Networks. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-9). IEEE.

[17] Orhei, C., Bogdan, V., Bonchis, C., & Vasiu, R. (2021). Dilated Filters for Edge-Detection Algorithms. Applied Sciences, 11(22), 10716.

[18] Elharrouss, O., Akbari, Y., Almaadeed, N., & Al-Maadeed, S. (2022). Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches. arXiv preprint arXiv:2206.08016.

[19] Y. Wang, X. Zhao, and K. Huang. Deep crisp boundaries. In CVPR, 2017.

[20] Jing, J., Liu, S., Wang, G., Zhang, W., & Sun, C. (2022). Recent advances on image edge detection: A comprehensive review. Neurocomputing.

[21] Bertasius, G., Shi, J., & Torresani, L. (2015). Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4380-4389).

[22] Cai, Y., Wang, Z., Luo, Z., Yin, B., Du, A., Wang, H., ... & Sun, J. (2020, August). Learning delicate local representations for multi-person pose estimation. In European Conference on Computer Vision (pp. 455-472). Springer, Cham.

[23] Deng, R., & Liu, S. (2020, October). Deep structural contour detection.

In Proceedings of the 28th ACM international conference on multimedia (pp. 304-312).

[24] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 5, pp. 530–549, 2004.

[25] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 5, pp. 898–916, 2011.

[26] P. Dollar and C. L. Zitnick, "Fast edge detection using structured ´forests," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 8, pp. 1558–1570, 2015.

[27] Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. In Proceedings of the IEEE international conference on computer vision (pp. 1395-1403).

[28] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In ECCV, pages 562–578, 2018.

[29] Park, Y. H., Seo, J., & Moon, J. (2020). Cafenet: class-agnostic few-shot edge detection network. arXiv preprint arXiv:2003.08235.

[30] Liu, Y., Cheng, M. M., Hu, X., Wang, K., & Bai, X. Richer Convolutional Features for Edge Detection, in IEEE Transactions on Pattern

Analysis and Machine Intelligence, vol. 41, no. 8, pp. 1939-1946, 1 Aug. 2019, doi: 10.1109/TPAMI.2018.2878849.

[31] Wang, L., Shen, Y., Liu, H., & Guo, Z. (2019). An accurate and efficient multi-category edge detection method. Cognitive Systems Research, 58, 160-172.

[32] He, Jianzhong, et al. "Bi-directional cascade network for perceptual edge detection." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[33] Sun, K., Xiao, B., Liu, D., Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 5693-5703).

[34] Le, T., & Duan, Y. (2020). REDN: a recursive encoder-decoder network for edge detection. IEEE Access, 8, 90153-90164.

[35] Su, Z., Liu, W., Yu, Z., Hu, D., Liao, Q., Tian, Q., ... & Liu, L. (2021). Pixel difference networks for efficient edge detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5117-5127).

[36] Al-Amaren, A., Ahmad, M. O., & Swamy, M. N. S. (2021). RHN: A Residual Holistic Neural Network for Edge Detection. IEEE Access, 9, 74646-74658.

[37] Li, O., & Shui, P. L. (2021). Color edge detection by learning classification network with anisotropic directional derivative matrices. Pattern Recognition, 118, 108004.

[38] Soria, X., Pomboza-Junez, G., Sappa, A. D. (2022). LDC: Lightweight Dense CNN for Edge Detection. IEEE Access, 10, 68281-68290.

[39] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[40] Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., & Shlens, J. (2019). Stand-alone self-attention in vision models. Advances in Neural Information Processing Systems, 32.

[41] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[42] Pu, M., Huang, Y., Liu, Y., Guan, Q., & Ling, H. (2022). EDTER: Edge Detection with Transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1402-1412).