**MultiResEdge: A Deep Learning-Based Edge Detection Approach**

Kanija Muntarina1,2, Rafid Mostafiz3, Fahmida Khanom4, Sumaita Binte Shorif1,

Mohammad Shorif Uddin1,\*

1Dept. of Computer Science and Engineering, Jahangirnagar University, Dhaka, Bangladesh

2Dept. of Computer Science and Engineering, Dhaka City College, Dhaka, Bangladesh

**3**Institute of Information Technology, Noakhali Science and Technology University, Noakhali, Bangladesh

4Department of Applied Mathematics, University of Dhaka, Dhaka 1000, Bangladesh

E-mail: muntarina123@gmail.com; rafid.iit@nstu.edu.bd; fahmida.dristy@gmail.com; sumaita.bs@gmail.com; shorifuddin@juniv.edu

\*Corresponding Author’s Email: shorifuddin@juniv.edu, ORCID: 0000-0002-7184-2809

**Abstract**

Edge detection and segmentation of target objects are very crucial in computer vision research to estimate the textural and structural details. Accurate pixel boundary mapping is an important research area that finds widespread applications in various fields, like image segmentation, shape extraction, pattern recognition, medical image processing, motion analysis, etc. In this study, we highlighted two primary limits that are faced in edge detection: edge connectivity and edge thickness. Choosing the right threshold for reliable edge detection has long been one of the most difficult problems in traditional edge detection techniques. To deal with these issues, the present research is motivated to execute a new deep learning-based edge detection model named *MultiResEdge* that is a tweaked variant of the original UNet framework in the multi-resolution environment. Instead of directly integrating encoder and decoder features, the suggested model makes use of a MutiRes block to modify spatial information from different scales, and a residual path with a set of convolutions to convey the features from the encoder to the decoder. To reduce the gap between the encoder and decoder features, we advocate using a combination of convolutional and skip connections. We also incorporated semantic edge information with semantic curves to improve the border. In addition, we have integrated a multi-step preprocessing technique that mitigates the false positives and false negatives and hence improves the accuracy, precision, recall, and F1-score. We also justify the predicted edge comparing it with the corresponding ground truth in terms of various objective metrics. Finally, we have compared our result with that obtained by the related state-of-the-art methodologies.

**Keywords**

Edge detection; Edge connectivity, Edge uniformity; Deep learning; UNet.

1. **Introduction**

In the past few decades, there has been a lot of research done on edge detection, which is a key step in many image processing and vision algorithms [1]. In the fields of image processing, computer vision, and robotics, the edge is a notable component that may be used for a wide variety of tasks, including segmentation, feature description, enhancement, pattern recognition, restoration, compression, and object tracking [2-9].

Edges are the areas in which there are discernible shifts in the intensity level of the underlying image, and they typically appear at the intersections of the contours of two distinct objects [10, 11]. The edge detection method refers to the process of finding the boundaries of objects within the image. In its most fundamental form, edge detection is a technique for subdividing an image into distinct parts of discontinuity [12]. An edge detector's function is to recognize, track down, and mark all abrupt discontinuities [13-15]. Edge detection reduces image data and maintains structural features for future processing.

There have been many different approaches to edge detection developed to extract edges from digital images. The algorithms for the identification of edges are based on i) classical technology and ii) deep learning technology.

Gradient-based operators of classical edge detection technology, such as Robert [16], Prewitt [17], and Sobel [18] were initially utilized for edge detection; however, these operators did not produce sharp edges and were extremely susceptible to noise in the image. The Laplacian, Gaussian-Laplacian, and Canny [19] operators also have two drawbacks: they have a high chance of identifying erroneous edges, and the localization error may be significant at curved edges. However, despite these drawbacks, the operators are still widely used. Edge detection using deep learning has become a vibrant concept because of developments in deep learning technology [20]. Deep learning-based edge detection technology differs from classical edge detection technology in that its characteristics, rather than being manually developed, are instead learned from substantial amounts of data.

The state-of-the-art techniques investigate the edge detections in three computer vision domains: (i) spatial domain: in this domain, an edge detector utilizes gradient-based operators of first-order type (such as Roberts, Prewitt, Sobel) and second-order type (such as Canny based on Laplacian and Gaussian smoothing) where the first order type works at the point of larger gradient value, but the second order type searches for the point of zero-crossing occur [21]; (ii) frequency domain: in this domain, an edge detector converts the incoming image signal into the frequency domain to determine the edge information [22]; (iii) wavelet domain: in this domain, an edge detector transforms the image into multifrequency levels using various sub-bands. However, noise affects gradient-based operators, such as (Roberts, Prewitt, and Sobel) greatly, and it is vulnerable to weak contour details [23]. Also, the main drawback of the Canny edge technique is the selection of a perfect threshold and therefore, causes confusion about associating pixels whether an edge or not an edge. Thus, it is observed that the majority of classical edge detection methods are noise-sensitive, somewhat area-based, and moment-based [24-26]. In addition, the most fundamental challenge associated with edge detection is the presence of uncertainty in the process of locating edge positions. This introduces the prospect of both false negatives (in which the process fails to identify actual edges) and false positives (non-edge detected as edge points). To address these situations, researchers are now inclined to practice deep-learning-oriented methods, as the neural features minimize the difficulties in edge detection and introduce better accuracy [27-32]. Recent CNN-based techniques have shown encouraging performance gains in the case of high precision and recall in edge detection. A new wave of research utilizing convolutional neural networks has also emphasized the value of autonomous hierarchical feature learning, such as DeepEdge [27], N4-Fields [28], CSCNN [29], and DeepContour [30]. One of the earliest efforts was to create an end-to-end edge detection system (holistically-nested edge detection (HED)) [31] that could automatically learn the rich hierarchical characteristics and resolve ambiguity in natural picture edge and object boundary recognition. Recently, [33] suggested image-to-image training and predictions similar to the HED merging all the convolutional layer outputs in an effective method by concatenating the features from the contracting path while conducting up-sampling in the expanding path using UNet architecture.

However, there remains a large room for improvement in these edge detection methods, both qualitatively (subjectively) and quantitively (objectively), and in edge connectivity and edge uniformity along with the detection accuracy. The following facts are a point-by-point summary of the edge detection issues that are causing performance degradation: (i) firstly, noise and flaws in the image make it difficult to locate actual edges in the correct locations. In addition, a noise suppressor linear filter produces a smoothing or blurring effect where there are abrupt changes in intensity; (ii) secondly, edge detectors typically produce crowded or sparse edges as a result of incorrect threshold selection. These edges typically comprise a large number of isolated edge points or broken segments that lack continuity and uniformity. This is a significant challenge for the implementation of computer vision tasks that need a brief and accurate depiction of the object boundaries.

In this research, we formulate an effective edge detection methodology where the interpolation is coupled with the contrast enhancement technique for intensities mapping on noise and artifact-eliminated smooth image to dig the potential edge information and preserve the sharp details. Then the dataset is trained and tested using the end-to-end Multi-CNN-based model consisting of a contracting path to capture context and an expanding path to precise edge localization. For the sake of an objective measure of a predicted edge, the proposed technique is evaluated with the recent existing deep learning and classical edge detection methods. We have compared the objective performance of the proposed method concerning recent methods through various metrics, such as Entropy, Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM), and Feature Similarity Index Metric (FSIM). The research contributions of this paper are summarized as follows.

* A deep learning model named MultiResEdge which is a tweaked version of the original UNet in the multi-resolution environment is proposed to identify structures maintaining edge connectivity and edge thickness in edge boundaries.
* The proposed framework resolves the drawbacks of traditional patch-based CNN methods to address unwanted blur and noise effects.
* Extensive experiments are performed based on bench-mark datasets that confirm effectiveness compared to state-of-the-art methodologies.

The remainder of this paper is organized as follows. Section 2 discusses the relevant research. Section 3 represents the research methodology in detail. Performance metrics along with result analyses are described in Section 4. Finally, this paper reaches the conclusion in section 5.

1. Relevant Literature

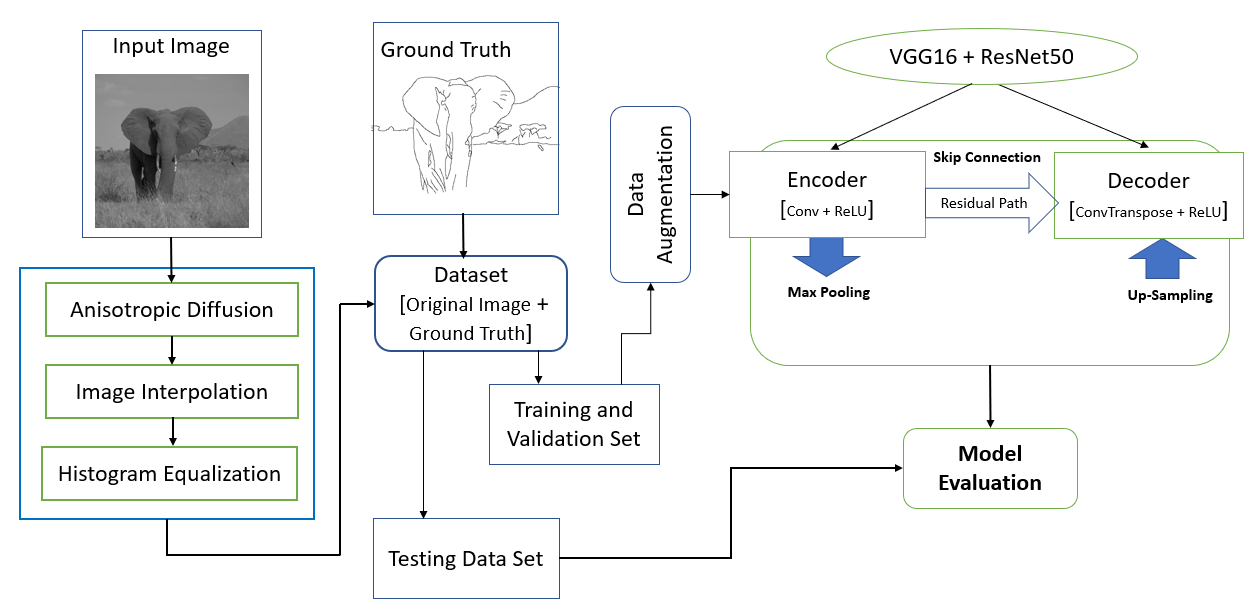
Researchers are driven to use deep learning-based methods to identify edges and object boundaries, as deep learning-based approaches produce amazing, superior, and more dynamic outcomes than standard approaches in a variety of fields [34-38]. In the field of image processing for identifying edges, Dollar et al. [39] attempted to utilize supervised learning in 2006 to determine the locations of the edges and bounds. In this adaptive learning stage, they selected and combined a considerable number of features from multiple scales. Their strategy used an extended model of the probabilistic boosting tree classification method for discriminative model learning. Arbelaez et al. [40] utilized a method to reduce the contour detection problem in image segmentation using spectral clustering-based globalization contour detection in conjunction with several local cue detections from a hierarchical region tree. Since 2015, deep-learning techniques gained greater acceptance in this field. Ou et al. [41] proposed contour detection utilizing the full convolution operation to semantic segmentation.

To execute edge point per-pixel classifications, Hwang and Liu [29] applied DenseNet, an effective multiscale convolutional neural network (CNN), to derive an informative feature vector for each image and then employed an SVM classifier to carry out edge identification. Bertasius et al. [27] set up a unified multiscale deep learning approach called DeepEdge based on KNet for edge prediction utilizing higher-level object knowledge. A CNN-based edge detection approach employing image patches to overcome additional feature extraction was demonstrated by Wang [42]. Xie and Tu [31] devised a revolutionary deep learning technique called HED (holistically nested edge detection) to resolve the shortcomings of canny edge detectors. A five-layer feature diversity-based richer convolution feature image edge detection extraction technique was suggested by Liu et al. [35]. Zhang et al. [43] implemented a rapid R-CNN (Region-based CNN) framework-based end-to-end edge-preserving neural network for pedestrian identification. A subdivided network for conspicuous target detection was presented by Zhao et al. [44]. Tian and Wei [45] proposed an anisotropic multiscale edge detection technique.

From this literature survey, it is evident that many methods are developed, and deep learning methods are gaining popularity as they are promising compared to traditional approaches. But the main problems of edge detection are to ensure edge boundaries for ensuring structures are still unsolved. Therefore, the current paper tries to address it by maintaining edge connectivity through a deep learning strategy termed ‘MultiResEdge.’

1. Research Methodology

This section demonstrates the working procedures of the proposed deep learning-based edge detection framework in detail and investigates the logical execution. This experiment follows a preprocessing technique that is robust against unwanted noise and artifacts. The edge-preserving anisotropic diffusion is used to eliminate the unnecessary Gaussian and speckle noises. After that, an interpolation coupling with the contrast enhancement mechanism is enabled to dig the potential details and rescue the input image to pass to the deep learning model for edge extraction. This model uses a multi-CNN backbone network, having an encoder to extract spatial features from the image and a decoder that constructs the segmentation map from the encoded features. The encoder follows the convolution operation of multi-CNN and the output from the convolution layer before the pooling layer for transferring to the decoder by introducing the concept of skip connection. This helps to retrieve the spatial information lost by the pooling layer. They are concatenated with the output from the up-sampling operations of the decoder. Figure 1 represents the block diagram of the proposed method.



**Fig. 1.** Block diagram of deep learning-based edge detection.

3.1 Edge detection principles and challenges

Edge is estimated by the intensity variation which is extracted using the first-order differentiation of the image function. In Fig. 2(a) the image function *f*(*x*, *y*) is visualized; Fig. 2(b) the intensity changes at the edge boundary along the x-axis at the middle of y-axis position for the given image *f*(*x*, *y*); Fig. 2(c) shows the boundary region in ground truth image. The variation in intensity is usually determined by the first-order differentiation of the image function. The derivative indicates high-intensity change at the edge boundary and low-intensity change at the homogeneous region.

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |
| Fig. 2. (a). Original image, (b) intensity distribution along the x-axis at the middle of y-axis position, (c) ground truth edge image. | | |

For a two-dimensional (2D) image function 𝑓(𝑥, 𝑦), edge strength can be represented by a gradient vector as

(1)

where *fx* and *fy* are thepartial derivatives of intensityalong with the *x*-axis and *𝑦*-axis directions, respectively. The gradient vector magnitude (which is the candidate edge) at a pixel location (𝑥, 𝑦) can be determined by L2 norm as

(2)

To reduce the computational burden, it can also be represented by L1 norm as

+ (3)

The image region is divided into object and non-object regions, which serve as a useful clue for object interpretation and recognition, as well as the edge boundary. The edge detection mechanism works by locating the intensity changes to extract the object boundaries. Modification of the true image intensity, however, is a regular occurrence due to various additive or multiplicative noises, imaging flaws, and uneven lighting conditions. As a result, an edge identification algorithm should include the following three core operations: smoothing (for noise reduction), differentiation (for creating candidate edges using image derivatives), and labeling (for locating actual edges by removing false edges) [46].

According to Zhu [47], edge detection needs to follow four requirements from the standpoint of computer vision. These requirements are that edges (i) be accurately detected, (ii) be in the right positions, (iii) form continuous lines (edge connectivity/continuity), and (iv) have a uniform width. Accuracy, connectivity/continuity, and homogeneity can be condensed into the three fundamental needs. The fundamental issue with edge detection is that detecting edge placements is unpredictable, which leaves room for both false positives (non-edge detected as edge points) and false negatives (failed to detect genuine edges).

3.2 Dataset and training parameters

The proposed edge detection mechanism is evaluated both qualitatively and quantitively using the Berkeley segmentation Dataset BSDS500 [40] and the Contour Image Dataset [48]. These two datasets are also manipulated by speckle and Gaussian effects, image flaws, and uneven contrast to prove the robustness of the proposed model. BSDS500 dataset includes 500 original images of dimension either (321×481) or (481×321) pixels along with annotated ground truth of the same dimension for each image. The Contour Image dataset contains 40 original images of size (512×512) pixels along the human-drawn ground truth of each image. From the BSDS500 dataset, randomly 80 percent (i.e, 400) images are used for training, and 20 percent (i.e, 100) images are used for testing. As the training image data are not sufficient, we increase this data by using an augmentation technique to handle the overfitting problem. For augmentation, we used zooming, rotation, shearing, width and height shifting, and horizontal and vertical flipping strategies in a Python environment. Table 1 shows these data augmentation strategies along with their parameters. After augmentation, the total number of images becomes 3200. Among these, randomly selected 2800 are used for pure training and the rest 400 are used for validation. For generalization, the Contour Image dataset is kept unseen and unknown and used for testing. Table 2 shows the details of the experimented datasets.

**Table 1.** Data augmentation strategies and parameters.

|  |  |
| --- | --- |
| **Data augmentation strategies** | **Parameter values** |
| Original image  Zooming range  Rotation range  Shearing range  Width shift range  Height shift range  Horizontal flip  Vertical flip | 224×224 pixels  2  90  0.5  0.4  0.4  True  True |

**Table 2.** Details of the experimented datasets.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dataset | Total original image | Training | | | | Testing |
| Original image | After Augmentation | Training | Validation | -- |
| Berkely BSDS500 | 500 | 400 | 3200 | 2800 | 400 | 100 |
| Contour Image | 40 | -- | -- | -- | | 40 |

The whole process of data manipulation, training, and testing tasks was executed in a Python environment. The framework is conducted on a computer with a memory of 24 GB, a GPU NVIDIA Quadro RTX 600, and a computation capacity of 7.5. In training, an extensive grid search is conducted to find the optimal parameters that give low-loss outcomes for the proposed MultiResEdge model. The best tuning parameters are listed in Table 3.

**Table 3.** Tuning Parameters.

|  |  |
| --- | --- |
| Parameter | Candidate selected |
| Optimizer  Learning rate  Learning rate steps  Learning rate  Batch size  Dropout probability  Activation function | ‘Adam’  1e-4  10  0.1  4  0.5  ‘sigmoid’ |

Here, the Adam optimizer is used for optimizing the learning rate. A dropout probability (*p*) of 0.5 was used during the model training to avoid overfitting [49] sigmoid activation function is used as the output of the multi-resolution UNet architecture is binary (i.e, 0 or 1). The loss function metric is given as:

(4)

Where,

(5)

are true and predicted levels, and *h* and *w* are image width and height, respectively. CE is the formula for cross-entropy between a true probability distribution *p* and an estimated probability distribution *p̂*. DL is the dice loss coefficient map between *p* and *p̂*.

3.3 Smoothing for noise reduction

This part introduces a multi-scale image preprocessing approach integrating three strategies: (i) edge-preserving anisotropic diffusion, (ii) interpolation, and (iii) adaptive histogram equalization. Anisotropic diffusion is a technique used in image processing for edge detection. The basic idea behind anisotropic diffusion is to smoothen an image in such a way that edges are preserved while noise is reduced. This is accomplished by diffusing the image over time, but in a way that favors diffusion in the direction perpendicular to the edges. Image interpolation is a technique used to increase or decrease the resolution of an image. It involves filling in the missing pixels in an image using different algorithms. By increasing the resolution of the image, we can potentially capture more subtle changes in intensity and therefore detect true edges. Adaptive Histogram Equalization (AHE) is a popular image enhancement technique that can be useful in edge detection applications. AHE improves the contrast of an image by redistributing the histogram of the image in a way that maximizes its dynamic range. This is particularly useful in cases of an image where image contrast is low and the edges are difficult to detect. Figs. 3(a) to 3(f) show images that are affected by various noises and blurring. In all cases, the sample image is enhanced through the above three preprocessing steps which are shown in Figs. 3(g) to 3(i). The final preprocessed image 3(i) of the sample will be applied to the proposed experiment.

|  |  |  |
| --- | --- | --- |
| 1. Gaussian effects | 1. Localvar effects | 1. Poisson effects |
| 1. Salt effects | 1. Paper effects | 1. Speckle effects |
| 1. Anisotropic diffusion | 1. Interpolation | 1. Histogram equalization |

**Fig. 3** (a) – (f) Images affected with various noises and blurring, (g) – (i) preprocessed image.

3.4 Deep learning model

The structure of the multi-CNN-based deep learning model – MultiResEdge (a tweaked UNet) is made up of a total eight blocks: four encoder blocks and four decoder blocks. Through a bridge, both the encoder and the decoder are connected. The encoder network takes the input image, which then travels through the encoder network. During this journey, the spatial dimension of the image cuts in half, and the number of filters is increased. In the decoder blocks, the number of filters is reduced while the sampling rate of the feature map is increased by a factor. The output of the very last decoder block is then sent to a 1×1 convolution that uses a sigmoid activation function as the final step. This results in the production of a binary segmentation mask for the input image. The segmented mask is compared then with the ground truth to evaluate the performance.

3.4.1 MultiRes block

To perform an analysis of objects across a variety of contexts, a framework like UNet using deep learning might prove useful. After each pooling operation and a transposed convolution in the initial structure of the UNet, a sequence of two convolutions with a kernel size of 3×3 was applied. This sequence of two convolutions of 3×3 was equivalent to a convolution of 5×5 [50]. As a result, the incorporation of 3×3 and 7×7 convolutions in parallel to the 5×5 convolution is the simplest way to enhance the UNet structure with a multi-resolution analysis while complying with the framework of the method of inception. Hence, by replacing the convolution operations with inception-like blocks, the architecture of UNet will have an easier way of feature adaptation that has been learned from images with a wide range of context sizes. There is also the possibility of making use of stride convolution operations [51]. Even though it improves performance, the introduction of additional parallel convolutional operations has the effect of vastly expanding the amount of memory that is necessary. As a result, it made use of a series of lightweight and smaller convolutions shown in Fig. 4(a) to factorize the more expensive and larger convolutions. It obtained the outputs of the three convolutions and then concatenated them together to extract the spatial features that were present at a variety of scales. This modification still requires a significant amount of memory, even though it significantly reduces the requirement for memory. The reason for this is that the number of filters present in the initial convolution has a quadratic influence on the memory if two convolutions are presented in a sequence within a deep neural structure. This constantly proliferated the filters for the three successive convolutional layers (from 1 to 3), rather than using the same number of filters in each of those layers, to prevent the memory requirement of the initial layers from propagating to the deeper section of the framework. This was done to stop the memory requirement from propagating. In addition, to ascertain additional spatial information, it included a residual connection for the implementation of 1×1 convolutions. Table 4 represents the configuration, which is referred to as a MultiRes block shown in Fig. 4(b).

|  |  |
| --- | --- |
| (a) | Left: Convolutional block used in the UNET architecture. Right: MultiRes block used in the MultiResUNET architecture.  (b) |

Fig. 4: (a) Convolutional Block; (b) MultiRes Block.

**Table 4.** MultiRes block details architecture.

|  |  |  |  |
| --- | --- | --- | --- |
| MultiRes Blocks | Layer | Size of filter | Number of filters |
| Block 1 | Convolution | 3×3 | 8 |
| Convolution | 3×3 | 17 |
| Block 9 | Convolution | 3×3 | 26 |
| Convolution | 1×1 | 51 |
| Block 2 | Convolution | 3×3 | 17 |
| Convolution | 3×3 | 35 |
| Block 8 | Convolution | 3×3 | 53 |
| Convolution | 1×1 | 105 |
| Block 3 | Convolution | 3×3 | 35 |
| Convolution | 3×3 | 71 |
| Block 7 | Convolution | 3×3 | 106 |
| Convolution | 1×1 | 212 |
| Block 4 | Convolution | 3×3 | 71 |
| Convolution | 3×3 | 142 |
| Block 6 | Convolution | 3×3 | 213 |
| Convolution | 1×1 | 426 |
| Block 5 | Convolution | 3×3 | 142 |
| Convolution | 3×3 | 284 |
| Convolution | 3×3 | 427 |
| Convolution | 1×1 | 853 |

3.4.2 Residual path

There is a semantic gap between the features of the encoder and the features of the decoder in the UNet architecture. The Residual Path was developed as a replacement for the simple skip connection of UNet. This condition has the potential to preserve the disintegrated spatial features that are otherwise lost during the pooling operation. However, there are issues with the skip connection such that the initial layers in the encoder part of the UNet model compute the low-level features and the deep layers in the decoder part of the model compute the notable high-level features, there may be a semantic gap between two collections of features that are being concatenated. After the most recent layer that was not sampled, the encoder and the decoder were joined together using the first skip connection to generate the initial pooling layer. As a consequence of this, the concatenation of these inconsistent collections of features has the potential to negatively influence the procedure for making predictions because they can cause inconsistencies during the process of learning. It was anticipated that the level of inconsistency would gradually decrease as it moves closer to the subsequent skip connections. This happened because the encoder features were not only concatenated with the features from the decoder part of the newer layers but were also moved with further processing. As a result, it has been suggested that the encoder and decoder both incorporate some convolutional operations along the skip connections to reduce the gap between their respective feature sets. Additionally, the implementation of a residual connection rather than making use of the conventional convolutional operation was chosen because this method produces an abundant number of deep structures and makes the process of learning easier [52]. This is performed by first passing the features through a sequence of convolutions and then merging them with the decoder features rather than simply merging the encoder and decoder features. It was anticipated that using these additional non-linear operations would result in a narrowing of the semantic gap that exists between the encoder and the decoder features. Table 5 depicts the suggested shortcut which is labeled as the Residual path, and the block structure is shown in Fig. 5. In particular, the 1×1 filters were used to accompany the residual connections, and the 3×3 filters were applied when convolutions were being performed.

**Table 5.** Residual path details architecture.

|  |  |  |  |
| --- | --- | --- | --- |
| Residual paths | Layer | Size of filter | Number of filters |
| Path 1 | Convolution | 3×3 | 64 |
| Convolution | 1×1 | 64 |
| Convolution | 3×3 | 64 |
| Convolution | 1×1 | 64 |
| Convolution | 3×3 | 64 |
| Convolution | 1×1 | 64 |
| Convolution | 3×3 | 64 |
| Convolution | 1×1 | 64 |
| Path 2 | Convolution | 3×3 | 128 |
| Convolution | 1×1 | 128 |
| Convolution | 3×3 | 128 |
| Convolution | 1×1 | 128 |
| Convolution | 3×3 | 128 |
| Convolution | 1×1 | 128 |
| Path 3 | Convolution | 3×3 | 256 |
| Convolution | 1×1 | 256 |
| Convolution | 3×3 | 256 |
| Convolution | 1×1 | 256 |
| Path 4 | Convolution | 3×3 | 512 |
| Convolution | 1×1 | 512 |

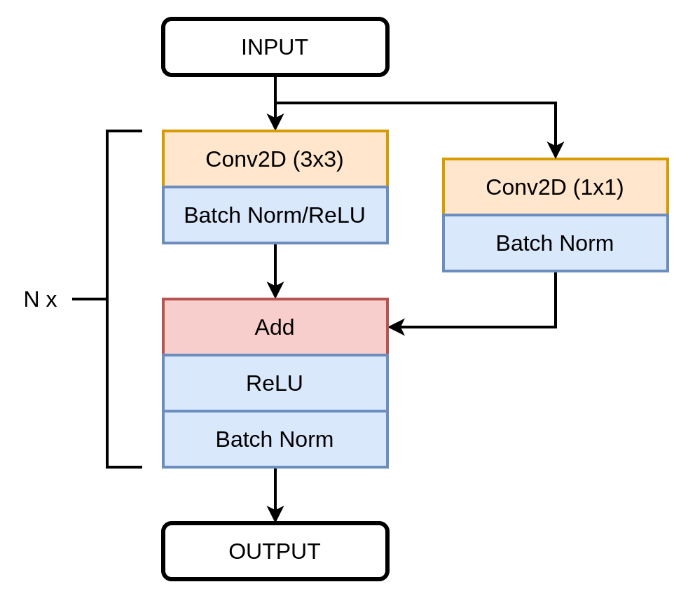


Fig. 5 Basic building structure of residual path.

3.4.3 MultiResEdge Architecture

The proposed MultiRes block is used in the multi-resolution environment instead of the series of two convolutions. A 𝑊 parameter is computed using equation (7) which has been allocated for every MultiRes block to control the number of filters of the convolutions inside the blocks.

(7)

Here, *U* defines the number of filters and α defines the scaler coefficient in the corresponding layer of UNet architecture.

The value of *W* grew doubled after each pooling or transposition of layers. It is decided to set a value of α=1.67 to keep the total number of parameters in our proposed network at a level that was slightly less than that of the UNet. It has been decided to use the metric *U* = [32, 64, 128, 256, and 512] to determine the number of filters that would be included in our proposed network. In addition to this, the filter values [*W*/6], [*W*/3], and [*W*/2] are assigned to each of the three subsequent convolutions, in the series. It is required that the number of filters in each subsequent convolution that takes place inside of a MultiRes block is increased. This should be done in place of the traditional method to keep the number of filters constant.

For integrating the encoder and decoder features, a new shortcut way (residual path) has been added and replaced with the usual skip connections employed in the original UNet model. For the feature maps that were disseminated from the contracting section (encoder) to the expanding part (decoder), a few different convolutional layers in the residual path were recommended to incorporate. Here the assumption is that the strength of the semantic gaps that existed between the encoder features maps and the decoder ones have been decreased via the internal shortcut pathways. As a consequence of this, the number of convolutional blocks that were used along each of the four Residual pathways gradually dropped until it reached 4, 3, 2, and 1. Also, to take into account, the number of feature maps in encoder–decoder, the filters are used with the sizes 32, 64, 128, and 256 in the four residual pathways. In the recommended network, the Rectified Linear Unit activation function (ReLU) was employed to activate all of the convolutional layers that were utilized, except for the output layer. In addition to that, each of them was normalized [53]. The architectural particulars of the recommended UNet in the multi-resolution environment are displayed in Table 4, and a representation of the model may be seen in Figure 6. To train our model, the binary cross entropy (BCE) function is calculated by utilizing the loss function equation (8).

Where, the original image, ground truth mask, and predicted mask from the proposed model are represented by respectively. Similarly, the image pixel is whose ground truth value is and the model prediction is . For the nth image batch, the loss function *J* is computed using equation (9).

(9)

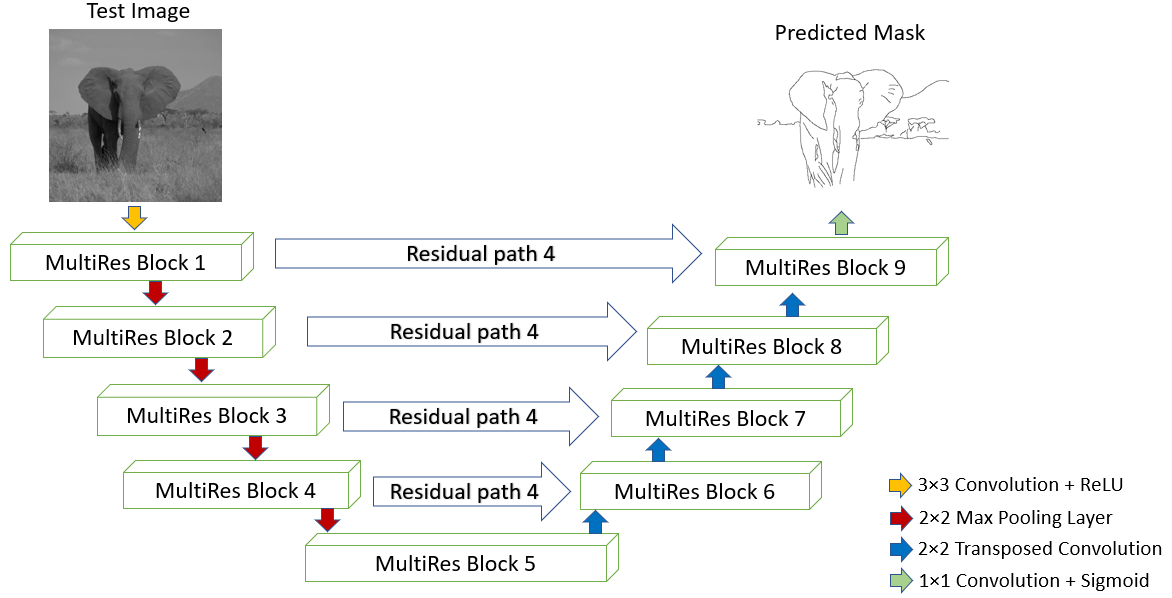


Fig. 6. MultiResEdge structure.

1. **Performance Evaluation and Experiment**

The performance of edge detection algorithms is evaluated both subjectively and objectively in this section. The human visual inspection evaluates the subjective measurement while objective evaluation is done by comparing the value of Entropy, MSE, PSNR, SSIM, and FSIM between the ground truth and the predicted images [54]. The objective evaluation results are listed in Table 6.

Table 6. Quantitative evaluation based on entropy, MSE, PSNR, SSIM, and FSIM.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Entropy | MSE | PSNR | SSIM | FSIM |
| DeepEdge [27] | 0.9479 | 0.2083 | 15.7025 | 0.3416 | 0.5062 |
| HED [31] | 0.5583 | 0.1514 | 18.6440 | 0.4841 | 0.7052 |
| FCN [32] | 0.7826 | 0.1604 | 17.2241 | 0.4365 | 0.6899 |
| UNet [33] | 0.4578 | 0.1268 | 19.2406 | 0.5242 | 0.7508 |
| Proposed MultiResEdge | 0.2951 | 0.0924 | 22.3368 | 0.8476 | 0.8980 |

Once the edges are detected, the entropy of the edges is determined by first computing the probability distribution of the edge pixel intensities. This can be done by counting the number of times each pixel intensity value occurs in the edge region. Next, compute the probability of each pixel intensity value by dividing its count by the total number of pixels in the edge region. Equation (10) is used to compute the entropy of the edges.

(10)

where is the entropy of the image, is the probability of occurrence of each pixel intensity value. A lower entropy value indicates that the edge region has lower uncertainty or randomness and is therefore more uniform, which is indicative of well-defined edges. Conversely, a higher entropy value indicates that the edge region is more uncertain and contains more noise or false edges.

To calculate the MSE, we first compute the difference between the detected edges and the ground truth edges at each pixel location. The difference is then squared and averaged over all pixel locations to obtain the MSE. Equation (11) is used for MSE computing.

(11)

where is the total number of pixels in the image, is the intensity value of the detected edge at the pixel , and is the intensity value of the ground truth edge at the pixel . A lower MSE value indicates better performance of the edge detection algorithm. However, it is important to note that the MSE may not always provide a complete evaluation of the algorithm's performance, as it only considers the pixel-level difference between the detected edges and the ground truth edges. Other metrics, such as precision, recall, and F1-score, should also be considered when evaluating the performance of an edge detection algorithm.

When it is talked about edge detection, we are referring to the process of identifying the boundaries between different regions of an image that have distinct textures, colors, or intensities. It is calculated as the ratio of the maximum possible power of a signal to the power of corrupting noise that affects the fidelity of its representation. Mathematically, equation (12) can define the PSNR of the edge of an image as follows.

(12)

where MSE is the mean squared error between the original edge and the degraded edge, and 255 is the maximum pixel value of an 8-bit grayscale image. The PSNR of the edge can be interpreted as a measure of the quality of the degraded edge image relative to the original edge image, with higher values indicating better quality. However, it is worth noting that PSNR is not always the best metric for evaluating image quality, as it does not always correlate well with human perception.

SSIM (Structural Similarity Index Measure) is a metric that is commonly used to evaluate the similarity between two images. It takes into account the luminance, contrast, and structure of the images, and is often used as a perceptual quality measure. One way to use SSIM in edge detection is to compare the edges detected by an algorithm with the ground truth edges in the original image. The SSIM score can be calculated for each edge, and the average score can be used as a measure of the overall quality of the edge detection algorithm. The SSIM index is calculated using equation (13).

(13)

Here, is the luminance component of the SSIM index, calculated as . and are the means of x and y, respectively, and is a small constant added for numerical stability, is the contrast component of the SSIM index, calculated as: . and are the variances of x and y, respectively; is the covariance of x and y, and is another small constant. is the structural component of the SSIM index, which measures the similarity of the local structures between x and y. It is calculated as: is the covariance of the local structures of x and y, and are the standard deviations of the local structures of x and y, respectively, and is another small constant. The SSIM index ranges from -1 to 1, with values closer to 1 indicating greater similarity between the two images. It is often used in image processing applications to evaluate the quality of image compression or restoration algorithms.

However, the FSIM (Feature Similarity Index Method) method provides a more accurate and reliable measure of edge similarity compared to traditional pixel-based methods. Next, the FSIM index is calculated based on the similarity between the filtered images. This is done by computing a similarity map that represents the degree of similarity between corresponding pixels in the two images. The similarity map is then weighted by a set of parameters that control the relative importance of different image features. FSIM is calculated by using equation (14) based on the Phase Congruency (PC) and as well as Magnitude Gradient (GM) and . The similarity of the predicted edge and ground-truth edge can be obtained using and .

(14)

The resulting FSIM index provides a measure of the similarity between the edge structures of the two images. A higher FSIM value indicates greater similarity between the edges, while a lower value indicates greater dissimilarity.

In this study, we selected some of the output samples of predicted edge photos from the test dataset to assess the performance of regularly used deep learning-based edge detectors named MultiResEdge, DeepEdge [27], HED [31], FCN [32] (fully convolutional neural networks), UNet [33], for visual evaluations. Figs. 7, 8, 9, and 10 show the comparative visual results of the subjective evaluation for the four samples from the BSD500 dataset. According to the deep learning-based analysis, the proposed MultiResEdge proves quite satisfactory performance in diverse domains of edge or boundary detection.

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |
| (d) | (e) | (f) |
| (g) |  |  |
| Fig.7 Visual results of the predicted edge of Sample-1 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge. | | |

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |
| (d) | (e) | (f) |
| (g) |  |  |
| Fig.8 Visual results of the predicted edge of Sample-2 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge. | | |

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |
| (d) | (e) | (f) |
| (g) |  |  |
| Fig.9 Visual results of the predicted edge of Sample-3 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge. | | |

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |
| (d) | (e) | (f) |
| (g) |  |  |
| Fig.10 Visual results of the predicted edge of Sample-4 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge. | | |

This experiment proposes a method for leveraging random pictures to analyze noise and artifact-affected images in low and high contrast. Several statistical parameters including accuracy, precision, recall, false-positive rate (FPR), true-negative rate (TNR), and F1-score are used to measure the efficiency and effectiveness quantitatively. Table 7 compares the proposed framework with the existing deep learning-based approaches that employed identical datasets. According to Table 7, the suggested method has the highest prediction accuracy in terms of edge detection when compared to earlier studies. By successfully applying the MultiResEdge model, we are able to achieve an average classification accuracy of 99.15% for the Berkely dataset and 97% for the Contour image dataset which is fully unseen and unknown and used only for generalization purposes. The equations (15), (16), (17), and (18) are used in deep learning modalities.

(15)

(16)

(17)

(18)

**Table 7.** Quantitative evaluation based on accuracy, precision, recall, and F1-score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score |
| Deep-Edge [27] | 0.94 | 0.95 | 0.92 | 0.93 |
| HED [31] | 0.97 | 0.97 | 0.97 | 0.97 |
| FCN [32] | 0.95 | 0.96 | 0.92 | 0.94 |
| UNet [33] | 0.97 | 0.98 | 0.97 | 0.97 |
| Proposed MultiResEdge | 0.99 | 0.98 | 0.98 | 0.98 |

High accuracy is the indicator of perfect prediction for the edge locations in the test outcome compared to the ground truth image. The minimum false positive rate means fewer unnecessary pixels in the unwanted regions which produce top precision values. At the same time, peak recall values prove the model’s minimum false negative rate. The F1-score is a useful metric in this experiment as there is an uneven distribution of classes in the dataset and one class may have many more instances than another. It provides a balanced measure of a model's ability to correctly identify both positive and negative instances. A higher F1-score indicates a better overall performance of the classification model.

The proposed model shows satisfactory performance with a precision of 0.98 and recall of 0.98, F1-score of 0.98 which outperforms the state-of-the-art methods. Fig. 11 shows the predicted edge maps of some samples from the Contour image dataset. These quite satisfactory results ensure the generalization capability of our proposed method.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
| 1. Original Image | 1. Ground Truth | 1. Predicted edge |

Fig.11 Predicted edge maps of some samples from the Contour image dataset using our proposed MultiResEdge.

The performance of usual edge detectors is discussed below:

Canny edge detection is one of the most widely used edge detection algorithms. It has high accuracy and low error rates. The F1 score for Canny edge detection ranges from 0.78 to 0.98, depending on the dataset and implementation. Sobel edge detection is a popular algorithm for real-time applications because of its simplicity and low computational cost. The F1 score for Sobel edge detection ranges from 0.6 to 0.9. Laplacian of Gaussian (LoG) edge detection is more robust to noise than other edge detection algorithms. The F1-score for LoG edge detection ranges from 0.5 to 0.95. Roberts edge detection is a simple and fast algorithm but is prone to noise. The F1 score for Robert’s edge detection ranges from 0.4 to 0.8. Prewitt edge detection is similar to Sobel edge detection but has a slightly lower accuracy. The F1 score for Prewitt edge detection ranges from 0.5 to 0.85. However, the performance of our proposed deep learning-based approach is quite high.

5. Conclusion

This research has executed a new deep learning-based edge detection framework - MultiResEdge to predict edge location in random images more precisely and reliably. The preprocessing techniques make this model more robust in case of any random artifact or unwanted blurring effects. This model is trained with variable contrast images so that the model can perform for generalization purposes also. The main contribution of this research is the low false negative rate and the low false positive rate which outperforms the all-other deep learning models as well as traditional techniques. Our proposed strategy may provide desirable outcomes from both normal and random photos, which is an improvement over the previous method. A method for analyzing additional evidence from the image and joining short contours into long ones is incorporated into the proposed system. Traditional algorithms are discussed for comparison. From the experimental results, we can draw the following conclusions: Firstly, the developed approach is capable of detecting strong and thin edges with a relatively low proportion of noise. Secondly, it offers higher-quality edge continuity. Thirdly, its entropy difference is lower from the ground truth and predicted edge image. Finally, we recommend that future work get some experience, if the proposed technique is used in real-time, improved object detection and recognition can be achieved. Robotic navigation and medical equipment can get benefit from the suggested edge detection technology.

**Data Availability**

The works used publicly available datasets mentioned in reference [40, 51].

**Funding**

No funding was received for this research.

**Conflict of interest**

The authors declare that they have no conflict of interest.

Authors' contributions

All authors have contributed equally and given consent to publish the manuscript.

References

1. K. Muntarina, S. B. Shorif, M. S. Uddin, “Notes on edge detection approaches,” Evolving Systems, vol. 13, no. 1, pp. 169–182, 2022.
2. W. Cui, G. Wu, R. Hua, H. Yang, “The research of edge detection algorithm for Fingerprint images,” in 2008 World automation congress, IEEE, 2008, pp. 1–5.
3. S. J. H. Pirzada and A. Siddiqui, “Analysis of edge detection algorithms for feature extraction in satellite images,” in 2013 IEEE International Conference on Space Science and Communication (IconSpace), IEEE, 2013, pp. 238–242.
4. H. S. Bhadauria, A. Singh, A. Kumar, “Comparison between various edge detection methods on satellite image,” International Journal of Emerging Technology and Advanced Engineering, vol. 3, no. 6, pp. 324–328, 2013.
5. S. M. Hou, C. L. Jia, Y. B. Wanga, M. Brown, “A review of the edge detection technology,” Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC), vol. 1, no. 2, pp. 26–37, 2021.
6. G. Rezai-Rad and M. Aghababaie, “Comparison of SUSAN and sobel edge detection in MRI images for feature extraction,” in 2006 2nd International Conference on Information & Communication Technologies, IEEE, 2006, pp. 1103–1107.
7. M. C. Shin, D. Goldgof, K. W. Bowyer, “An Objective Comparison Methodology of Edge Detection Algorithms Using a Structure from Motion Task,” Proc. 1998 IEEE CVPR, Santa Barbara, CA, 1998, pp. 190-195.
8. K. X. Yang and M. H. Sheu, “Edge-based moving object tracking algorithm for an embedded system,” 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Jeju, South Korea, 25-28 October 2016, pp. 153-155.
9. S. K. Pal and R. A. King, “On edge detection of x-ray images using fuzzy sets,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 5, no. 1, pp. 69–77, 1983.
10. D. Marr and E. Hildreth, “Theory of edge detection,” Proceedings of the Royal Society of London. Series B. Biological Sciences, vol. 207, no. 1167, pp. 187–217, 1980.
11. E. C. Hildreth, “The detection of intensity changes by computer and biological vision systems,” Computer Vision, Graphics, and Image Processing, vol. 22, no. 1, pp. 1–27, 1983.
12. M. Kumar and R. Saxena, “Algorithm and technique on various edge detection: A survey,” Signal & Image Processing, vol. 4, no. 3, p. 65, 2013.
13. X. Wang, “Laplacian operator-based edge detectors,” IEEE Transactions on pattern analysis and machine intelligence, vol. 29, no. 5, pp. 886–890, 2007.
14. R. M. Haralick, “Digital step edges from zero-crossings of second directional derivatives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, no. 1, pp. 58–68, 1984.
15. V. S. Nalwa and T. O. Binford, “Ondetecting edges,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 699–714, 1986.
16. S. K. Nair, S. K. Chinnappan, A. K. Dubey, A. Subburaj, S. Subramaniam, V. Balasubramaniam, S. Sengan, “Prewitt Logistic Deep Recurrent Neural Learning for Face Log Detection by Extracting Features from Images,” Arabian Journal for Science and Engineering, pp. 1–12, 2021.
17. C. Lyu, Y. Chen, A. Alimasi, Y. Liu, X. Wang, J. Jin, “Seeing the Vibration: Visual-Based Detection of Low Frequency Vibration Environment Pollution,” IEEE Sensors Journal, vol. 21, no. 8, pp. 10073–10081, 2021.
18. R. V. Patil and Y. P. Reddy, “An autonomous technique for multi class weld imperfections detection and classification by support vector machine,” Journal of Nondestructive Evaluation, vol. 40, no. 3, p. 76, 2021.
19. Y. D. Pranata, K. C. Wang, J. C. Wang, I. Idram, J. Lai, J. W. Liu, I. Hsieh, “Deep learning and SURF for automated classification and detection of calcaneus fractures in CT images,” Computer methods and programs in biomedicine, vol. 171, pp. 27–37, 2019.
20. M. Mittal, A. Verma, I. Kaur, B. Kaur, M. Sharma, L. M. Goyal, S. Roy, T. H.Kim, “An efficient edge detection approach to provide better edge connectivity for image analysis,” IEEE Access, vol. 7, pp. 33240–33255, 2019.
21. R. M. Haralick, “Digital step edges from zero-crossings of second directional derivatives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, no. 1, pp. 58–68, 1984.
22. K. S. Shanmugam, F. M. Dickey, J. A. Green, “An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 1, no. 1, pp. 37-49, 1979.
23. F. Han, B. Liu, J. Zhu, B. Zhang, “Algorithm design for edge detection of high-speed moving target image under noisy environment,” Sensors, vol. 19, no. 2, p. 343, 2019.
24. G.-S. Xu, “Sub-pixel edge detection based on curve fitting,” in 2009 Second International Conference on Information and Computing Science, IEEE, 2009, pp. 373–375.
25. A. Trujillo-Pino, K. Krissian, M. Alemán-Flores, D. Santana-Cedrés, “Accurate subpixel edge location based on partial area effect,” Image and Vision Computing, vol. 31, no. 1, pp. 72–90, 2013.
26. S. Ghosal and R. Mehrotra, “Orthogonal moment operators for subpixel edge detection,” Pattern Recognition, vol. 26, no. 2, pp. 295–306, 1993.
27. G. Bertasius, J. Shi, L. Torresani, “Deepedge: A multi-scale bifurcated deep network for top-down contour detection,” in Proceedings of the IEEE Conference on computer vision and pattern recognition, 2015, pp. 4380–4389.
28. Y. Ganin and V. Lempitsky, “N4-fields: neural network nearest neighbor fields for image transforms,” in Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II, Springer, 2015, pp. 536–551.
29. J. J. Hwang and T. L. Liu, “Pixel-wise deep learning for contour detection,” arXiv preprint arXiv:1504.01989, 2015.
30. W. Shen, X. Wang, Y. Wang, X. Bai, Z. Zhang, “Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3982–3991.
31. S. Xie and Z. Tu, “Holistically-nested edge detection,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1395–1403.
32. J. Long, E. Shelhamer, T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
33. O. Ronneberger, P. Fischer, T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.
34. Z. Yu, C. Feng, M.-Y. Liu, S. Ramalingam, “Casenet: Deep category-aware semantic edge detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5964–5973.
35. Y. Liu, M. M. Cheng, X. Hu, K. Wang, X. Bai, “Richer convolutional features for edge detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3000–3009.
36. Y. Liu, M. M. Cheng, D. P. Fan, L. Zhang, J. W. Bian, D. Tao, “Semantic edge detection with diverse deep supervision,” International Journal of Computer Vision, vol. 130, no. 1, pp. 179–198, 2022.
37. Y. Liu and M. S. Lew, “Learning relaxed deep supervision for better edge detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 231–240.
38. S. Sarkar, V. Venugopalan, K. Reddy, J. Ryde, N. Jaitly, M. Giering, “Deep learning for automated occlusion edge detection in RGB-D frames,” Journal of Signal Processing Systems, vol. 88, pp. 205–217, 2017.
39. P. Dollar, Z. Tu, S. Belongie, “Supervised learning of edges and object boundaries,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), IEEE, 2006, pp. 1964–1971.
40. P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, “Contour detection and hierarchical image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 5, pp. 898–916, 2010.
41. Y. Ou, X. He, S. Qu, “Fully convolutional neural network with attention module for semantic segmentation,” Journal of Frontiers of Computer Science and Technology, vol. 16, no. 5, pp. 1136–1145, 2022.
42. R. Wang, “Edge detection using convolutional neural network,” in Advances in Neural Networks–ISNN 2016: 13th International Symposium on Neural Networks, ISNN 2016, St. Petersburg, Russia, July 6-8, 2016, Proceedings 13, Springer, 2016, pp. 12–20.
43. H. Zhang, Y. Du, S. Ning, Y. Zhang, S. Yang, C. Du, “Pedestrian detection method based on Faster R-CNN,” in 2017 13th International Conference on Computational Intelligence and Security (CIS), IEEE, 2017, pp. 427–430.
44. F. Zhao, W. Zhang, Z. Yan, H. Yu, W. Diao, “Multi-feature map pyramid fusion deep network for semantic segmentation on remote sensing data,” Journal of Electronics & Information Technology, vol. 41, no. 10, pp. 2525–2531, 2019.
45. B. Tian and W. Wei, “Research Overview on Edge Detection Algorithms Based on Deep Learning and Image Fusion,” Security and Communication Networks, 2022.
46. D. Ziou and S. Tabbone, “Edge detection techniques-an overview,” Pattern Recognition and Image Analysis, vol. 8, pp. 537–559, 1998.
47. Q. Zhu, “Efficient evaluations of edge connectivity and width uniformity,” Image and Vision Computing, vol. 14, no. 1, pp. 21–34, 1996.
48. C. Grigorescu, N. Petkov, M. A. Westenberg, “Contour detection based on nonclassical receptive field inhibition,” IEEE Transactions on Image Processing, vol. 12, no. 7, pp. 729–739, 2003.
49. R. Mostafiz, M. S. Uddin, I. Jabin, M. M. Hossain, M. M. Rahman, “Automatic Brain Tumor Detection From MRI Using Curvelet Transform and Neural Features,” International Journal of Ambient Computing and Intelligence (IJACI), vol. 13, no. 1, pp. 1–18, 2022.
50. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
51. P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, G. Cottrell, “Understanding convolution for semantic segmentation,” in 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, pp. 1451–1460.
52. C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, “Inception-v4, Inception-ResNet and the impact of residual connections on learning,” in Proceedings of the AAAI conference on artificial intelligence, 2017.
53. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” International Conference on Machine Learning, 2015, pp. 448–456.
54. U. Sara, M. Akter, M. S. Uddin, “Image quality assessment through FSIM, SSIM, MSE and PSNR - A comparative study,” Journal of Computer and Communications, Vol. 7, pp. 8-18, March 2019.