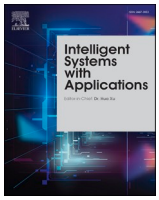
Intelligent Systems with Applications 20 (2023) 200274

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications



MultiResEdge: A deep learning-based edge detection approach

Kanija Muntarina [a,b], Rafid Mostafiz [c], Fahmida Khanom [d,e], Sumaita Binte Shorif [a], Mohammad Shorif Uddin [a,*]

[a] Department of Computer Science and Engineering, Jahangirnagar University, Dhaka, Bangladesh
[b] Department of Computer Science and Engineering, Dhaka City College, Dhaka, Bangladesh
[c] Institute of Information Technology, Noakhali Science and Technology University, Noakhali, Bangladesh
[d] Department of Applied Mathematics, University of Dhaka, Dhaka 1000, Bangladesh
[e] Department of Computer Science and Engineering, Green University of Bangladesh, Bangladesh

ARTICLE INFO

Keywords:
Edge detection
Edge connectivity
Edge uniformity
Deep learning
Multi-resolution UNet

ABSTRACT

Edge detection is a fundamental technique in image processing and computer vision that plays a crucial role in various applications including object recognition, shape analysis, segmentation, feature extraction, image enhancement, image understanding, compression, and preprocessing. It involves identifying the boundaries or edges between different objects or regions which enables us to extract valuable visual information, analyze image structure, and facilitate subsequent computer vision tasks. In this study, we highlighted two primary issues 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* which is a tweaked variant of the original *UNet* framework. 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. The extensive experiment is performed using images from two benchmark datasets to justify the predicted edge comparing it with the corresponding ground truth in terms of various objective metrics (such as entropy, MSE, PSNR, SSIM, and FSIM). Overall, our proposed edge detection model has obtained an accuracy of 99% along with an F1 score of 98% which is better than the related state-of-the-art methodologies.

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 (Muntarina et al., 2022). 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 (Bhadauria et al., 2013; Cui et al., 2008; Hou et al., 2021; Pal & King, 1983; Pirzada & Siddiqui, 2013; Rezai-Rad & Aghababaie, 2006; Shin et al., 1998; Yang & Sheu, 2016). 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 (Hildreth, 1983; Marr & Hildreth, 1980). 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 (Kumar & Saxena, 2013). An edge detector's function is to recognize, track down, and mark all abrupt discontinuities (Haralick, 1984; Nalwa & Binford, 1986; Wang, 2007). Edge detection reduces image data and maintains structural features for future processing.

Many approaches to edge detection have been 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.

* Corresponding author.
E-mail address: shorifuddin@juniv.edu (M.S. Uddin).

https://doi.org/10.1016/j.iswa.2023.200274
Received 25 April 2023; Received in revised form 19 August 2023; Accepted 29 August 2023
Available online 3 September 2023

Gradient-based operators of classical edge detection technology, such as Robert (Nair et al., 2021), Prewitt (Lyu et al., 2021), and Sobel (Patil & Reddy, 2021) 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 (Pranata et al., 2019) operators also have two drawbacks: they have a high chance of identifying erroneous edges, and the localization error may be significant at curved edges. 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 (Haralick, 1984); (ii) frequency domain: in this domain, an edge detector converts the incoming image signal into the frequency domain to determine the edge information (Shanmugam et al., 1979); (iii) wavelet domain: in this domain, an edge detector transforms the image into multi-frequency 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 (Han et al., 2019). 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 (Ghosal & Mehrotra, 1993; Trujillo-Pino et al., 2013; Xu, 2009). 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).

Dealing with noise and other unwanted variations in the image (illumination, size, and orientation), handling various edge types (such as step edges, ramp edges, and roof edges), determining the proper edge thresholds or criteria, and striking a balance between detecting all relevant edges and preventing false detections are some of the major challenges in edge detection. The following facts are a point-by-point summary of the edge detection problems that are causing performance degradation: (i) 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) 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 to identify structures maintaining edge connectivity and edge thickness in edge boundaries for various computer vision tasks including navigation that need a brief and accurate depiction of the object boundaries. To address these issues of edge detection, researchers are now inclined to practice deep-learning-oriented methods, as the neural features minimize the difficulties in edge detection and introduce better accuracy (Bertasius et al., 2015; Ganin & Lempitsky, 2015; Hwang & Liu, 2015; Long et al., 2015; Shen et al., 2015; Xie & Tu, 2015). 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.

Recently, CNN (convolutional neural network)-based techniques have shown encouraging performance gains in the case of high precision and recall in edge detection through autonomous hierarchical feature learning, such as DeepEdge (Bertasius et al., 2015), N4-Fields (Ganin & Lempitsky, 2015), CSCNN (Hwang & Liu, 2015), and DeepContour (Shen et al., 2015). One of the earliest efforts was to create an end-to-end edge detection system (holistically-nested edge detection (HED)) (Xie & Tu, 2015) that could automatically learn the rich hierarchical characteristics and resolve ambiguity in natural picture edge and object boundary recognition. Recently, image-to-image training and predictions suggested 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 (Ronneberger et al., 2015).

To improve accuracy, we formulate an effective edge detection methodology where the interpolation is coupled with the contrast enhancement technique for intensity 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.

2. Relevant literature

Many scholars devoted their attention to developing efficient edge operators. The related works regarding related literature have been included in Table 1 by summarizing their strengths, limitations, and methods, in a meta-analysis tabular format.

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 (Liu & Lew, 2016; Liu et al., 2022; Liu et al., 2017; Sarkar et al., 2017; Yu et al., 2017). In the field of image processing for identifying edges, Ou et al. (2022) proposed contour detection utilizing the full convolution operation to semantic segmentation. Hwang and Liu (2015) 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. (2015) 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 (2016). Xie & Tu (2015) devised a revolutionary deep learning technique called HED (holistically nested edge detection) to resolve the shortcomings of canny edge detectors. Zhang et al. (2017) 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. (2019). Tian and Wei (2022)

Table 1
Related works regarding edge detection techniques.

Paper	Methodology	Outcomes/strengths	Limitations
Abdou and Pratt (1979)	A thresholding/enhancement edge detection technique.	Obtain a variety of small and large mask edges.	False edge detection for the noise and artifact-affected images.
Kitchen and Rosenfeld (1981)	Find the edges based on the local good form.	It does not require prior knowledge of genuine locations.	The lack of width homogeneity.
Zhu (1996)	Statistical analysis of the edge pixel patterns identified in the images after an edge detection and/or linking operation.	Quantitative measurements of edge continuity and width uniformity.	Do not measure the edge locations and the ratio of true edges and non-edges
Ghita and Whelan (2002)	Exponential function to remove image noise and analyzing local information around edge terminators.	Determine the linking path in the unconnected edge structure and edge linking algorithm produces quality connected edge maps.	Low computational speed.
Russo (1998)	Adopted a fuzzy reasoning for the successful detection of edges without being misled by the noises.	Extract edges from data corrupted by impulse noise	Computation time.
Gao et al. (2010)	An improved Sobel edge detector using the soft-threshold wavelet de-noising.	Edge detection on images that include White Gaussian noises.	False positive rate and some blurring effects.
El-Khamy et al. (2000)	Obtains four threshold values and applies fuzzy reasoning concepts to enhance the edges of the image.	Attempted a very clear and better edge contour	Rigorous objective procedure of evaluation was not performed
Dollar et al. (2006)	An expanded version of the Probabilistic Boosting Tree classification method which picked and combined a large number of characteristics from multiple scales.	Identify edge and boundary based on supervised learning.	It failed in showing high performance in the Berkeley dataset which contains natural images
ArbelÆez et al. (2011)	Uses a spectral clustering-based globalization contour detection in conjunction with several local cue detections	Created a segmentation method with the intention of employing general machinery to convert the results of contour detection into a hierarchical area tree	Demanded fixed-resolution images for the appropriate detection of object boundaries
Lim et al. (2013)	A novel mid-level feature termed sketch token, which was supervised in the form of manually drawn contours in image data.	Random forest classifier for object recognition from photos to cluster the contour patches.	Human-generated contour patches
Agarwal and Goel (2016)	Evaluated the edge detection of the canny approach with the bacterial foraging algorithm (BFA).	BFA outperformed Canny because it employs swarm intelligence	Problem in driving connection between the local phase vector and the local attenuation.
Yang et al. (2016)	A modified differential phase congruency technique was created to assume that a higher dimensional signal is not necessarily a one-dimensional signal.	A higher dimensional space frequency in that approach is equivalent to the negative of the scale derivate of the local attenuation	The conventional shortcomings.
Xuan and Hong (2017)	Differential operation on amplitude gradient histograms	Separate the targets from their background and was noise-resistant.	The weak edge information.
Verma and Parihar (2017).	A fuzzy approach for edge identification using the shortest unvalued segment assimilating nucleus (SUSAN) principle and BFA	A parametric fuzzy intensifier operator (FINT) was created to enhance the weak edge information.	Noisy picture.
Günen et al. (2017)	The backtracking search clustering technique.	An edge identification technique for noisy pictures.	Did not work on extra noisy picture.
Ma et al. (2018)	(SAR) image detection algorithm based on the de-noising algorithm via the sparse representation and a new morphology edge detector.	Effective in edge positioning accuracy, integrity, and the number of false edge points.	Calculation speed.
Zhang et al. (2018)	Utilized FPGA (Field Programmable Gate Array) technology in to enhance the Sobel edge technique.	Increase the gradient template and parallel processing capability with high reliability, flexibility and reconfigurability.	No future scope.
Cao et al. (2018)	Otsu-optimized Canny operator using a MapReduce parallel programming model that runs on the Hadoop platform.	To address speed and cost to determine its dual thresholds.	In real-time large-scale image edge detection processing.

proposed an anisotropic multiscale edge detection technique. Some real-life applications of edge detections are presented in the Refs. Aye et al. (2019), Karen et al. (2006), Öztürk et al. (2006), Yildiz et al. (2003).

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.'

3. Materials and method

This section demonstrates the working procedures (system model) of the proposed deep learning-based edge detection framework in detail and investigates the logical execution. This experiment follows a pre-processing 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. Fig. 1 represents a block diagram (flowchart) of the proposed method.

3.1. Mathematical model of edge detection

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-

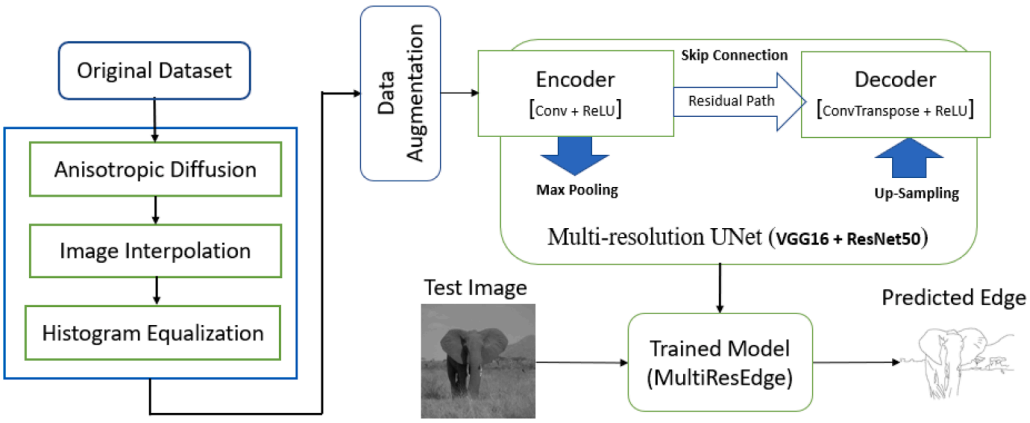


Fig. 1. Block diagram of deep learning-based edge detection.

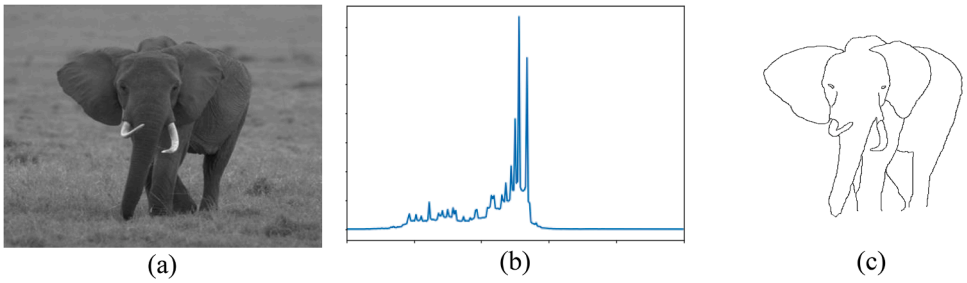


Fig. 2. (a). Original image, (b) intensity distribution along the x-axis at the middle of y-axis position, (c) ground truth edge image.

intensity change at the edge boundary and low-intensity change at the homogeneous region.

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

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\nabla f}{\nabla y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{1}$$

where f_x and f_y are the partial derivatives of intensity along 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 L_2 norm as

$$|\nabla f| = \sqrt{f_x^2 + f_y^2} \tag{2}$$

To reduce the computational burden, it can also be represented by L_1 norm as

$$|\nabla f| = |f_x| + |f_y| \tag{3}$$

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) (Ziou & Tabbone, 1998).

According to Zhu (1996), 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 (ArbelÆez et al., 2011) and the Contour Image Dataset (Grigorescu et al., 2003). 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% (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 2 shows these data augmentation strategies along with their parameters. After augmentation, the total number of images becomes 3200. Among these, randomly selected 2800 images are used for pure training and the rest 400 images are used for validation. For generalization, the Contour Image dataset is kept unseen and unknown and used for testing. Table 3 shows the details of the experimented datasets.

The whole process of data manipulation, training, and testing tasks was executed in a Python environment. The framework is conducted on

Table 2
Data augmentation strategies and parameters.

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

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 4.

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 (Szegedy et al., 2016) 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:

$$Loss = 0.5 \times mean(CE(p,\hat{p})) + 0.5 \times DL(p,\hat{p}) \tag{4}$$

Where,

$$CE(p,\hat{p}) = -(p\log(\hat{p}) + (1-p)\log(1-\hat{p})) \tag{5}$$

$$DL(p,\hat{p}) = 1 - \frac{2\sum_w\sum_h p.\hat{p}}{\sum_w\sum_h p + \sum_w\sum_h \hat{p}} \tag{6}$$

p and $\hat{p}$ 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 $\hat{p}$. DL is the dice loss coefficient map between p and $\hat{p}$.

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. Fig. 3(a)–(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 Fig. 3(g)–(i). The final preprocessed image 3(i) of the sample will be applied to the proposed experiment.

3.4. Deep learning model

The structure of the multi-CNN-based deep learning model multi-resolution UNet is made up of a total of 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 edge for the input image. The segmented edge 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 (Szegedy et al., 2016). 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 (Wang et al., 2018). 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 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

Table 4
Tuning Parameters.

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

Table 3
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

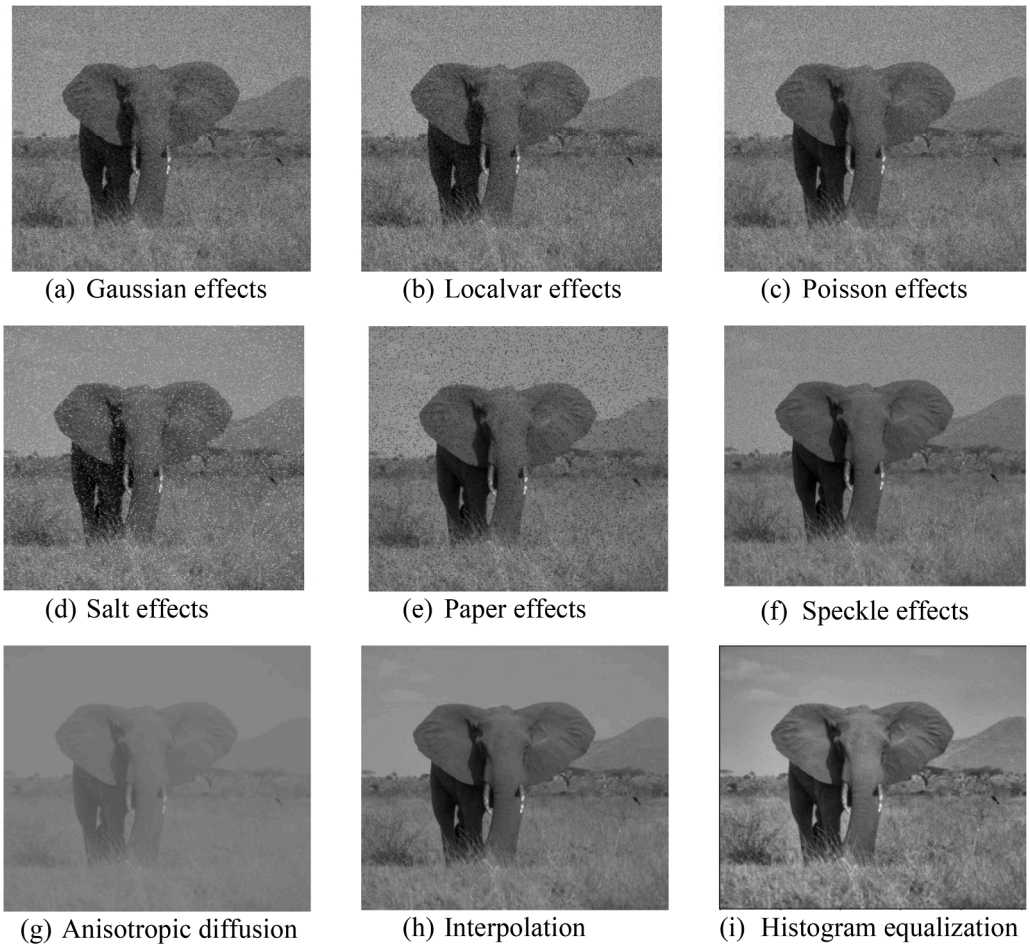


Fig. 3. (a)–(f) Images affected with various noises and blurring, (g) – (i) preprocessed image.

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 5 represents the configuration, which is referred to as a MultiRes block. Fig. 4 represents the Multires block compare to the basic convolutional block.

Table 5
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
	Convolution	3×3	142
Block 5	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

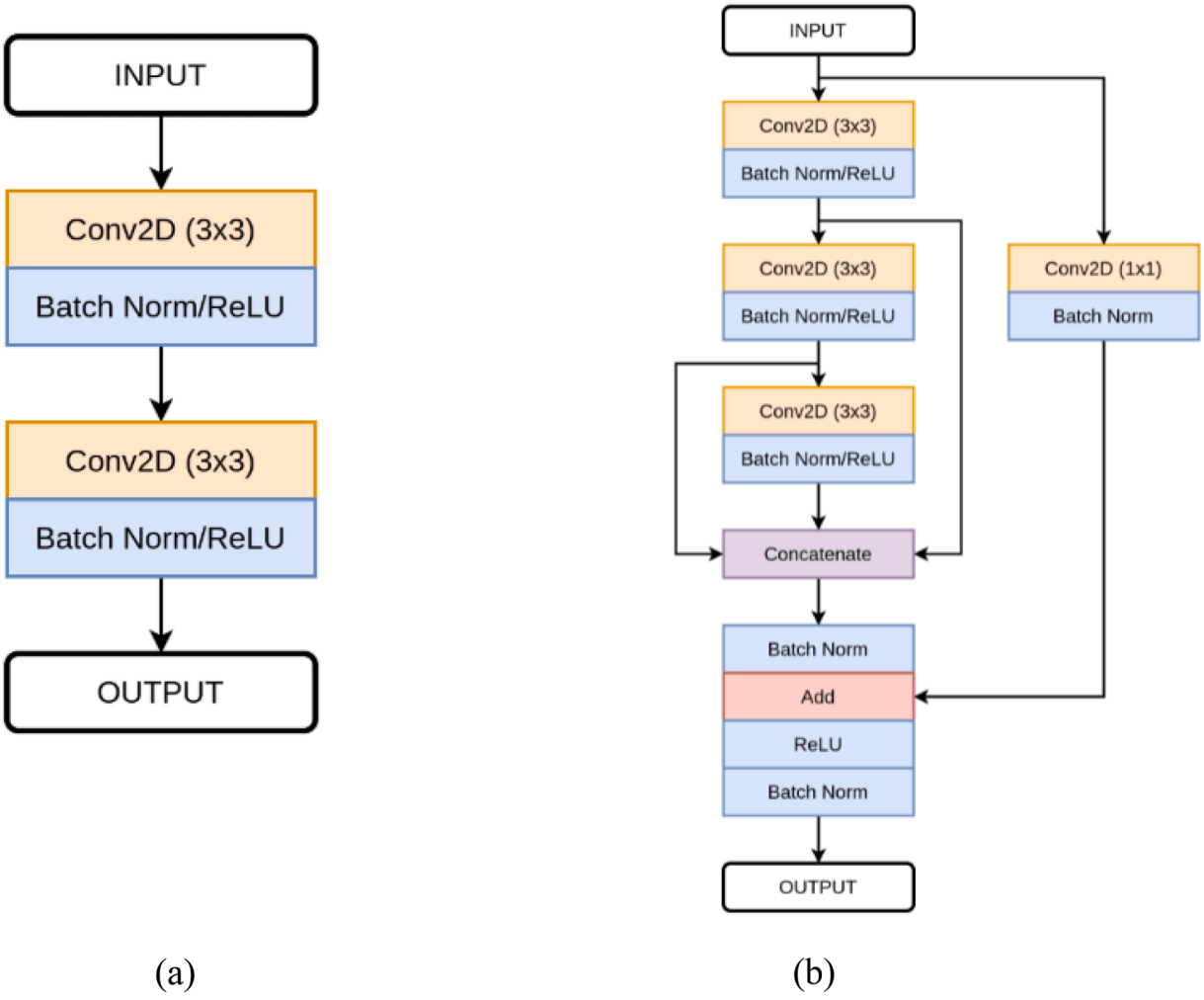


Fig. 4. (a) Convolutional Block; (b) MultiRes Block.

easier (Szegedy et al., 2017). 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 6 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.

3.4.3. *Multi-resolution UNet architecture*

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

$$W = \alpha \times U \tag{7}$$

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 $U = [32, 64, 128, 256, \text{and } 512]$ 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

Table 6
Residual path details architecture.

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

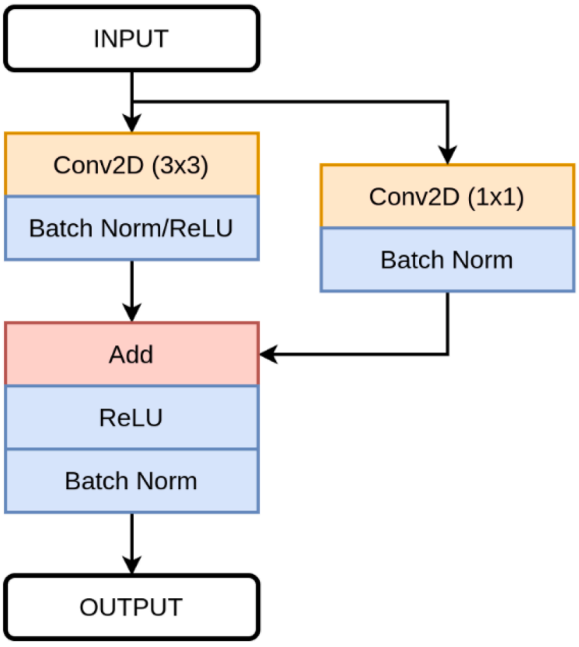


Fig. 5. Basic building structure of residual path.

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 blocks. 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 (Grigorescu et al., 2003). The architectural particulars of the recommended multi-resolution UNet network are displayed in Table 4, and a representation of the network may be seen in Fig. 6. To train the multi-resolution UNet model, the binary cross entropy (BCE) function is calculated by utilizing the loss function Eq. (8).

$$BCE(X, Y, \widehat{Y}) = \sum_{px \in X} -\left(y_{px} \log\left(\widehat{y}_{px}\right) + \left(1 - y_{px}\right) \log\left(1 - \widehat{y}_{px}\right)\right) \tag{8}$$

Where, the original image, ground truth, and predicted edge from the proposed model are represented by $X, Y, \widehat{Y}$, respectively. Similarly, the image pixel is px whose ground truth value is y_{px} and the model prediction is $\widehat{y}_{px}$. For the nth image batch, the loss function J is computed using Eq. (9).

$$J = \frac{1}{n} \sum_{i=1}^{n} BCE(X, Y, \widehat{Y}) \tag{9}$$

The details working procedure is presented in Algorithm 1.

4. Experiment results and discussion

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 Eqs. (10)–(13) are used in deep learning modalities.

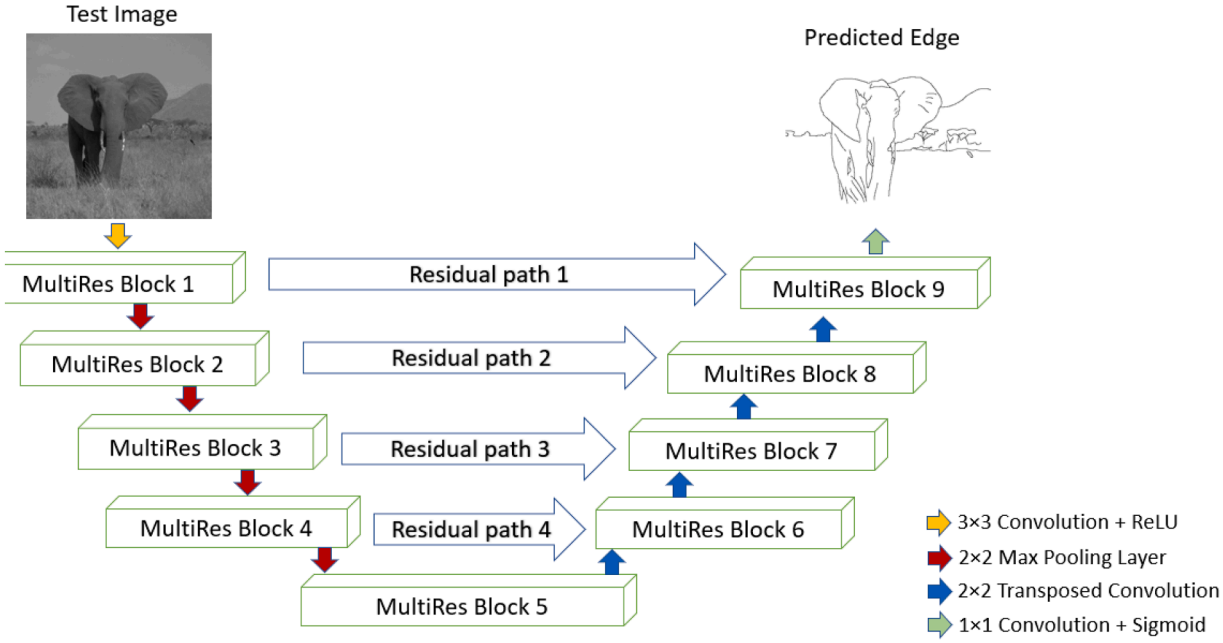


Fig. 6. Multi-resolution UNet structure.

Algorithm 1
MultiResEdge detection algorithm.

Input: Preprocessed images with corresponding ground truth
Output: Predicted edge
Begin
lowerRoman%1 Do the three preprocessing steps on each input image:
a Anisotropic Diffusion
a Interpolation
a Histogram Equalization
i Do data augmentation to increase the data variation and size.
i For the 9 MultiResBlocks:

a MultiResBlock→1:4; encode features from two convolutions of filter size: 3 × 3.

a MultiResBlock→2:5; add max-pooling of filter size: 2 × 2.

a MultiResBlock→5; encode features from two convolutions of filter size: 3 × 3 and then decode by the convolutions of filter size: 3 × 3 and 1 × 1.

a MultiResBlock→6:9; each block is decoded by the convolutions of filter size: 3 × 3 and 1 × 1. After that, add a transpose convolution of filter size: 2 × 2 in each block.

a Residual Paths:
- ResidualPath→4; concatenate one convolution of filter size: 3 × 3 with one convolution of filter size: 1 × 1.
- ResidualPath→3; concatenate two times one convolution of filter size: 3 × 3 with one convolution of filter size: 1 × 1.
- ResidualPath→2; concatenate three times one convolution of filter size: 3 × 3 with one convolution of filter size: 1 × 1.
- ResidualPath→1; concatenate four times one convolution of filter size: 3 × 3 with one convolution of filter size: 1 × 1.

End

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

Model	Accuracy	Precision	Recall	F1-score
Deep-Edge (Bertasius et al., 2015)	0.94	0.95	0.92	0.93
HED (Xie & Tu, 2015)	0.97	0.97	0.97	0.97
FCN (Long et al., 2015)	0.95	0.96	0.92	0.94
UNet (Ronneberger et al., 2015)	0.97	0.98	0.97	0.97
Proposed MultiResEdge	0.99	0.98	0.98	0.98

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

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 FS 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. 7 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.

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. The objective evaluation results are listed in Table 8.

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. Eq. (14) is used to compute the entropy of the edges.

$$H = -\sum_{i=0}^{255} p_i \log_2 p_i \tag{14}$$

where H is the entropy of the image, p_i 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. Eq. (15) is used for MSE computing.

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x,y) - \widehat{g}(x,y)^2 \tag{15}$$

where MN is the total number of pixels in the image, $g(x,y)$ is the intensity value of the detected edge at the pixel (x,y), and $\widehat{g}(x,y)$ is the intensity value of the ground truth edge at the pixel (x,y). 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, Eq. (16) can define the PSNR of the edge of an image as follows.

$$PSNR = 20\log_{10} \frac{255}{\sqrt{MSE}} \tag{16}$$

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 Eq. (17).

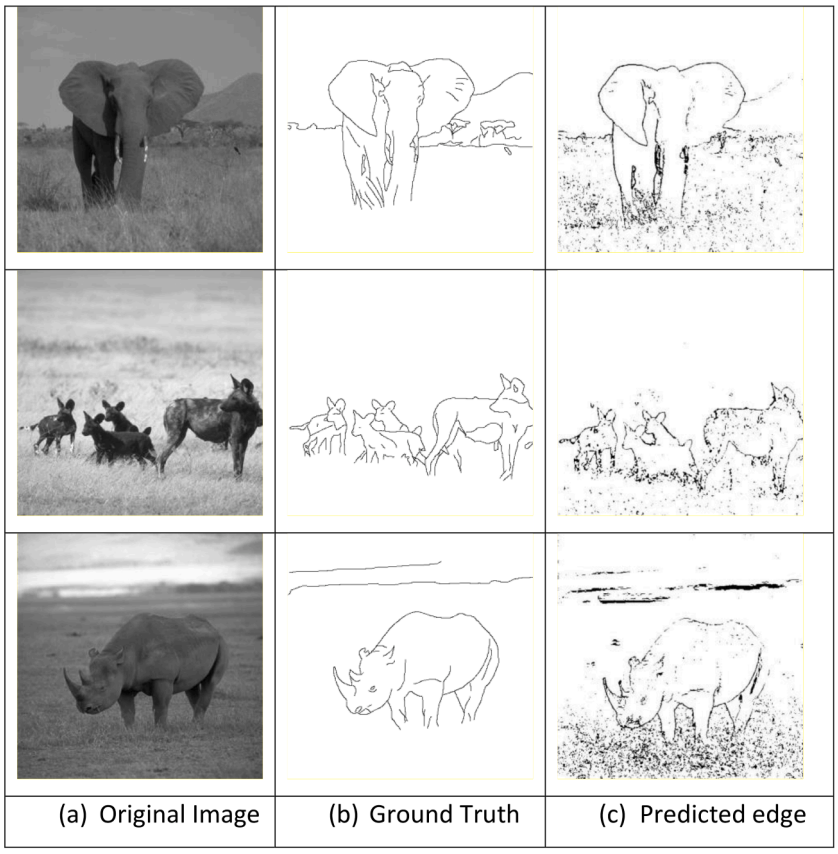


Fig. 7. Predicted edge maps of some samples from the Contour image dataset using our proposed MultiResEdge.

Table 8
Quantitative evaluation based on entropy, MSE, PSNR, SSIM, and FSIM.

Model	Entropy	MSE	PSNR	SSIM	FSIM
DeepEdge (Bertasius et al., 2015)	0.9479	0.2083	15.7025	0.3416	0.5062
HED (Xie & Tu, 2015)	0.5583	0.1514	18.6440	0.4841	0.7052
FCN (Long et al., 2015)	0.7826	0.1604	17.2241	0.4365	0.6899
UNet (Ronneberger et al., 2015)	0.4578	0.1268	19.2406	0.5242	0.7508
Proposed MultiResEdge	0.2951	0.0924	22.3368	0.8476	0.8980

$$SSIM = [I(x,y)]^{\alpha} \times [c(x,y)]^{\beta} \times [s(x,y)]^{\gamma} \tag{17}$$

Here, $I(x,y)$ is the luminance component of the SSIM index, calculated as $I(x,y) = \frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}$. μ_x and μ_y are the means of x and y, respectively, and C_1 is a small constant added for numerical stability, $c(x,y)$ is the contrast component of the SSIM index, calculated as: $c(x,y) = \frac{2\sigma_{xy}+C_2}{\sigma_x^2+\sigma_y^2+C_2}$. σ_x^2 and σ_y^2 are the variances of x and y, respectively; σ_{xy} is the covariance of x and y, and C_2 is another small constant. $s(x,y)$ is the structural component of the SSIM index, which measures the similarity of the local structures between x and y. It is calculated as: $s(x,y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$. σ_{xy} is the covariance of the local structures of x and y, σ_x and σ_y are the standard deviations of the local structures of x and y, respectively, and C_3 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 Eq. (18) based on the Phase Congruency (PC) PC_1 and PC_2 as well as Magnitude Gradient (GM) G_1 and G_2. The similarity of the predicted edge and ground-truth edge can be obtained using $S_{PC} = \frac{2PC_1.PC_2+T_1}{PC_1^2+PC_2^2+T_1}$ and $S_G = \frac{2G_1.G_2+T_2}{G_1^2+G_2^2+T_2}$.

$$FSIM = [S_{PC}(x,y)]^{\alpha}.[S_G(x,y)]^{b} \tag{18}$$

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 (Bertasius et al., 2015), HED (Xie & Tu, 2015), FCN (Long et al., 2015) (fully convolutional neural networks), UNet (Ronneberger et al., 2015), for visual evaluations. Figs. 8–11 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.

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

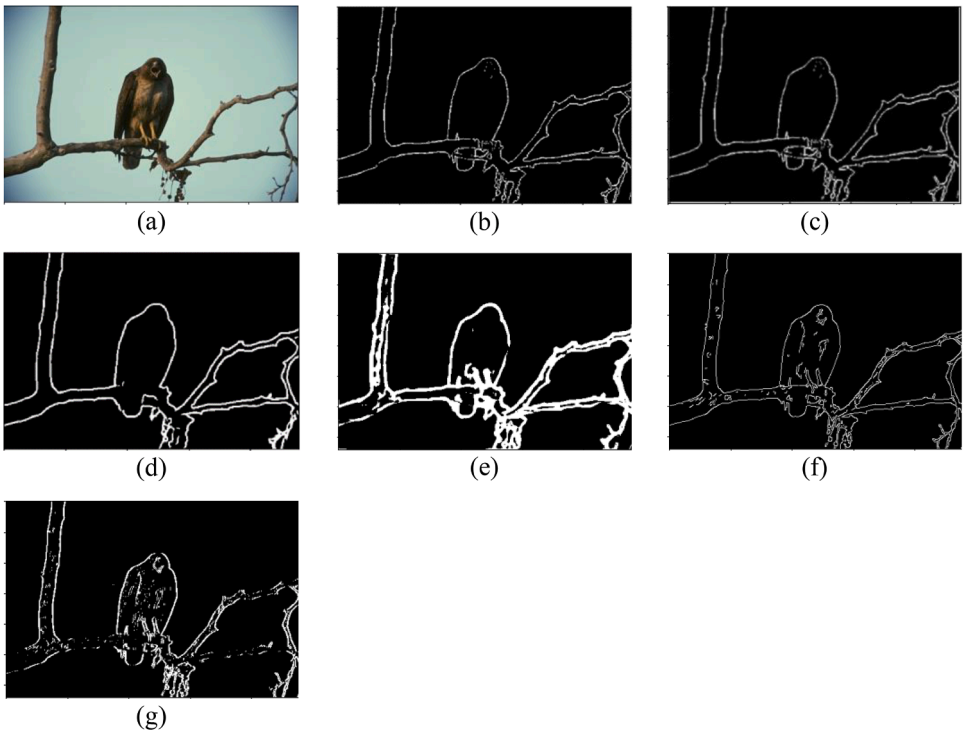


Fig. 8. Visual results of the predicted edge of Sample-1 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge.

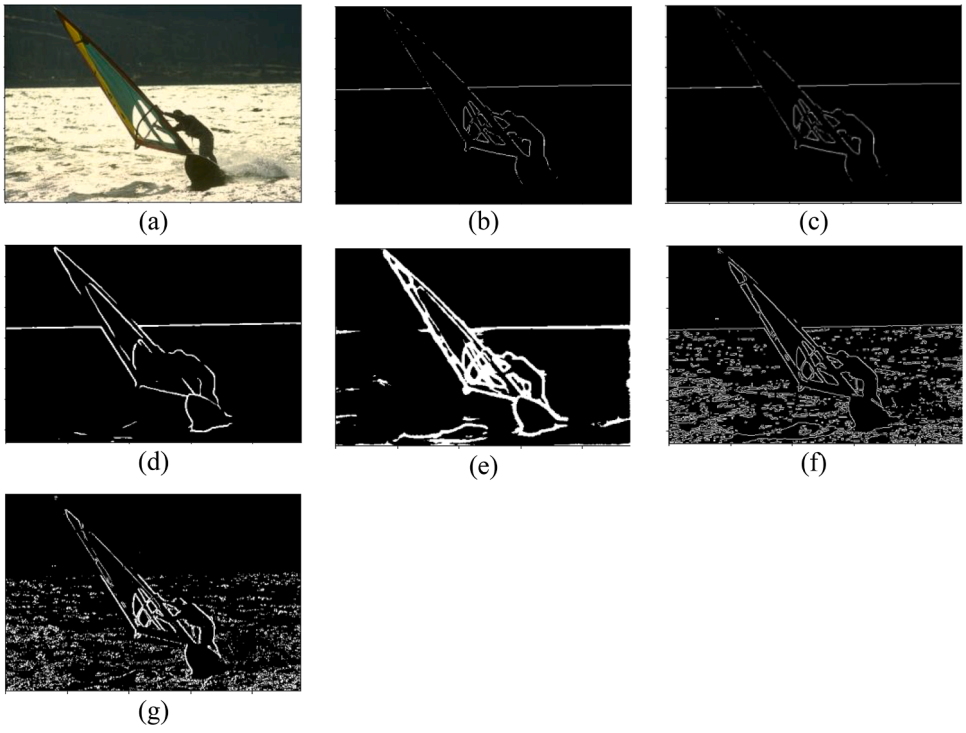


Fig. 9. Visual results of the predicted edge of Sample-2 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge.

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.

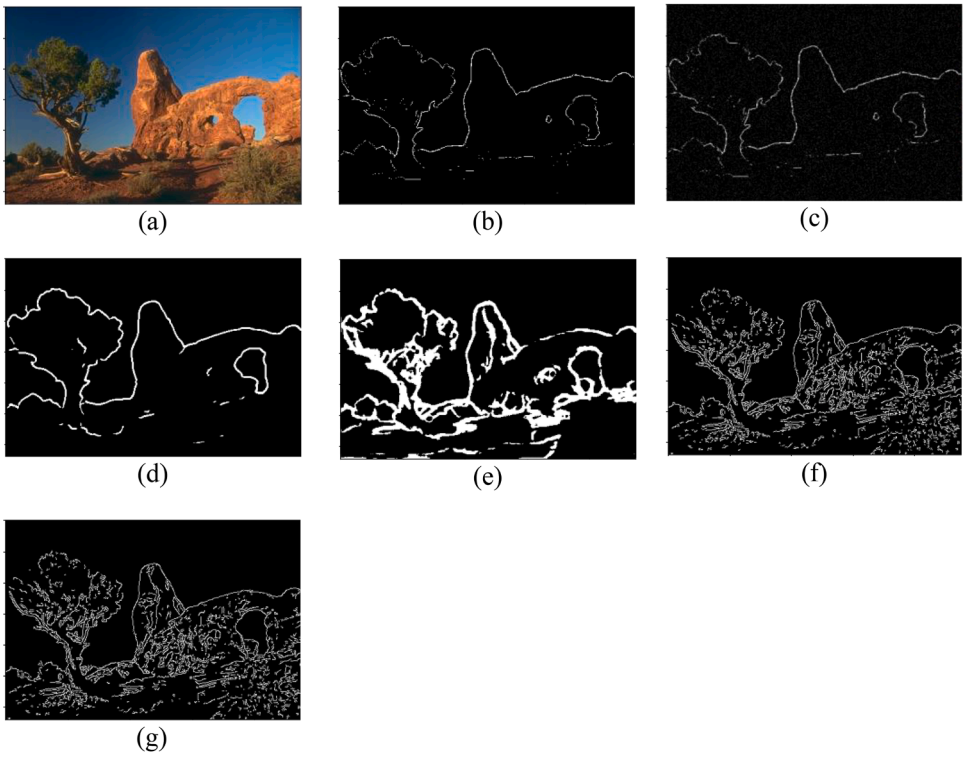


Fig. 10. Visual results of the predicted edge of Sample-3 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge.

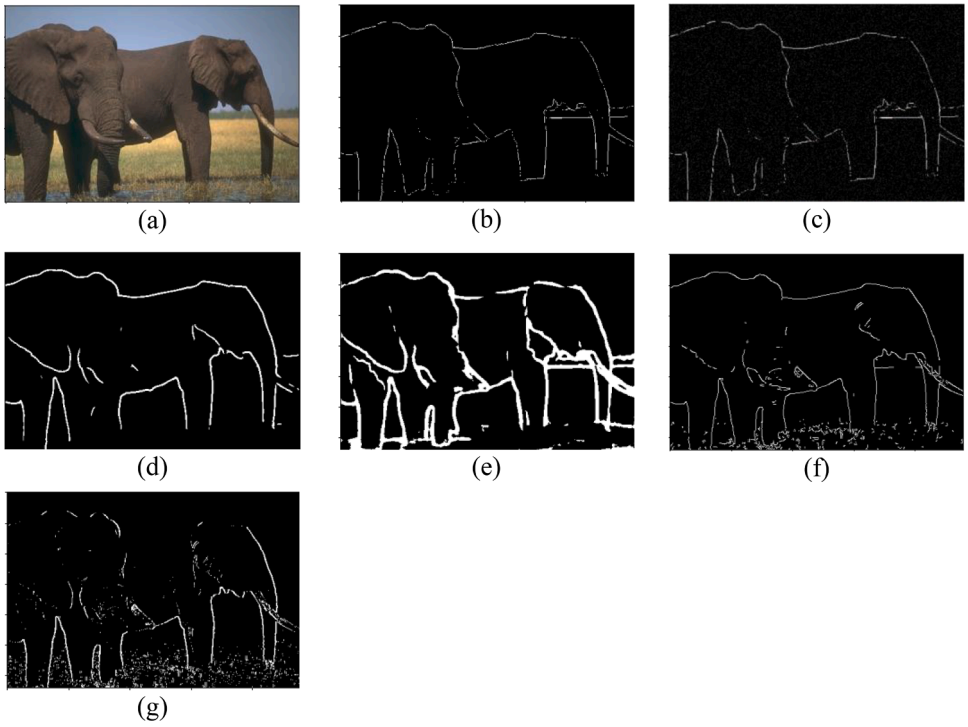


Fig. 11. Visual results of the predicted edge of Sample-4 from the BSD500 dataset: (a) Input test image, (b) Original ground-truth, (c) Proposed MultiResEdge, (d) UNet, (e) HED, (f) FCN, (g) DeepEdge.

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. Our proposed edge detection model has obtained an accuracy of 99% along with an F1 score of 98% which is better than the related state-of-the-art methodologies.

Besides, we can draw the following conclusions from our experimental results: 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. Future work may focus on real-time edge detection, which will find wide applications in robots as well as autonomous vehicle navigation and medical equipment. This model produces some false detection in the case of occluded pixels and future work will try to propose a generative model to fix this problem.

Data availability

The works used publicly available datasets mentioned in Refs. ArbelÆez et al. (2011) and Grigorescu et al. (2003).

Funding

No funding was received for this research.

Authors' contributions

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

CRediT authorship contribution statement

Kanija Muntarina: Conceptualization. **Rafid Mostafiz:** Methodology, Writing – original draft. **Fahmida Khanom:** Methodology, Writing – original draft. **Sumaita Binte Shorif:** Writing – review & editing. **Mohammad Shorif Uddin:** Conceptualization, Writing – review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no conflict of interest.

Data availability

Publicly available datasets are used.

References

Abdou, I. E., & Pratt, W. K. (1979). Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proceedings of the IEEE, 67*(5), 753–763. May.

Agarwal, A., & Goel, K. (2016). 'Comparative analysis of digital image for edge detection by using bacterial foraging & canny edge detector. In *Proceedings of the 2nd international conference computer intelligent* (pp. 125–129). Commun. Technol., Feb.

ArbelÆez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). 'Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 33*(5), 898–916. May.

Aye, C. M., Pholdee, N., Yildiz, A. R., Bureerat, S., & Saiz, S. M. (2019). Multi-surrogate-assisted metaheuristics for crashworthiness optimisation. *International Journal of Vehicle Design, 80*(2–4), 223–240.

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).

Bhadauria, H. S., Singh, A., & Kumar, A. (2013). Comparison between various edge detection methods on satellite image. *International Journal of Emerging Technology and Advanced Engineering, 3*(6), 324–328.

Cao, J., Chen, L., Wang, M., & Tian, Y. (2018). Implementing a parallel image edge detection algorithm based on the Otsu-canny operator on the Hadoop platform. *Computational Intelligence and Neuroscience, 2018*, Article 3598284. MaArt.

Cui, W., Wu, G., Hua, R., & Yang, H. (2008). The research of edge detection algorithm for Fingerprint images. In *Proceedings of the world automation congress* (pp. 1–5). IEEE.

Dollar, P., Tu, Z., & Belongie, S. (2006). Supervised learning of edges and object boundaries. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 1–8). Jun.

El-Khamy, S. E., Lotfy, M., & El-Yamany, N. (2000). A modified fuzzy Sobel edge detector. In *Proceedings of the 7th national radio science conference* (pp. 1–9). Feb.

Ganin, Y., & Lempitsky, V. (2015). N4-fields: Neural network nearest neighbor fields for image transforms. In *Proceedings of the computer vision–ACCV 2014: 12th Asian conference on computer vision* (pp. 536–551). Springer. November 1-5, 2014, Revised Selected Papers, Part II.

Gao, W., Zhang, X., Yang, L., & Liu, H. (2010). An improved Sobel edge detection. In *Proceedings of the 3rd international conference computer science information technology (ICCSIT)* (pp. 67–71). Jul.

Ghita, O., & Whelan, P. F. (2002). Computational approach for edge linking. *Journal of Electronic Imaging, 11*(4), 479.

Ghosal, S., & Mehrotra, R. (1993). Orthogonal moment operators for subpixel edge detection. *Pattern Recognition, 26*(2), 295–306.

Grigorescu, C., Petkov, N., & Westenberg, M. A. (2003). Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing, 12*(7), 729–739.

Günen, M. A., Atasever, Ü. H., & Be‡dok, E. (2017). A novel edge detection approach based on backtracking search optimization algorithm (BSA) clustering. In *Proceedings of the 8th international conference information technology (ICIT)* (pp. 116–122). Ma.

Han, F., Liu, B., Zhu, J., & Zhang, B. (2019). Algorithm design for edge detection of high-speed moving target image under noisy environment. *Sensors, 19*(2), 343.

Haralick, R. M. (1984). Digital step edges from zero-crossings of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 6*(1), 58–68.

Hildreth, E. C. (1983). The detection of intensity changes by computer and biological vision systems. *Computer Vision, Graphics, and Image Processing,, 22*(1), 1–27.

Hou, S. M., Jia, C. L., Wanga, Y. B., & Brown, M. (2021). A review of the edge detection technology. *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC), 1*(2), 26–37.

Hwang, J. J., & Liu, T. L. (2015). "Pixel-wise deep learning for contour detection," arXiv preprint arXiv:1504.01989.

Karen, I., Yildiz, A. R., Kaya, N., Öztürk, N., & Oeztuerk, F. (2006). Hybrid approach for genetic algorithm and Taguchi's method based design optimization in the automotive industry. *International Journal of Production Research, 44*(22), 4897–4914.

Kitchen, L., & Rosenfeld, A. (1981). Edge evaluation using local edge coherence. *IEEE Transactions on Systems, Man, and Cybernetics, 11*(9), 597–605. Sep.

Kumar, M., & Saxena, R. (2013). Algorithm and technique on various edge detection: A survey. *Signal & Image Processing, 4*(3), 65.

Lim, J. J., Zitnick, C. L., & DollÆr, P. (2013). Sketch tokens: A learned midlevel representation for contour and object detection. In *Proceedings of the IEEE conference computer vision pattern recognition* (pp. 3158–3165). Jun.

Liu, Y., Cheng, M. M., Fan, D. P., Zhang, L., Bian, J. W., & Tao, D. (2022). Semantic edge detection with diverse deep supervision. *International Journal of Computer Vision, 130*(1), 179–198.

Liu, Y., Cheng, M. M., Hu, X., Wang, K., & Bai, X. (2017). Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3000–3009).

Liu, Y., & Lew, M. S. (2016). Learning relaxed deep supervision for better edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 231–240).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Lyu, C., Chen, Y., Alimasi, A., Liu, Y., Wang, X., & Jin, J. (2021). Seeing the vibration: Visual-based detection of low frequency vibration environment pollution. *IEEE Sensors Journal, 21*(8), 10073–10081.

Ma, X., Liu, S., Hu, S., Geng, P., Liu, M., & Zhao, J. (2018). SAR image edge detection via sparse representation. *Soft Computing, 22*(8), 2507–2515.

Marr, D., & Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London Series B Biological Sciences, 207*(1167), 187–217.

Muntarina, K., Shorif, S. B., & Uddin, M. S. (2022). Notes on edge detection approaches. *Evolving Systems, 13*(1), 169–182.

Nair, S. K., Chinnappan, S. K., Dubey, A. K., Subburaj, A., Subramaniam, S., Balasubramaniam, V., et al. (2021). Prewitt logistic deep recurrent neural learning for face log detection by extracting features from images. *Arabian Journal for Science and Engineering*, 1–12.

Nalwa, V. S., & Binford, T. O. (1986). Ondetecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 8*(6), 699–714.

Ou, Y., He, X., & Qu, S. (2022). Fully convolutional neural network with attention module for semantic segmentation. *Journal of Frontiers of Computer Science and Technology, 16*(5), 1136–1145.

Öztürk, N., Yıldız, A. R., Kaya, N., & Öztürk, F. (2006). Neuro-genetic design optimization framework to support the integrated robust design optimization process in CE. *Concurrent Engineering, 14*(1), 5–16.

Pal, S. K., & King, R. A. (1983). On edge detection of x-ray images using fuzzy sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 5*(1), 69–77.

Patil, R. V., & Reddy, Y. P. (2021). An autonomous technique for multi class weld imperfections detection and classification by support vector machine. *Journal of Nondestructive Evaluation, 40*(3), 76.

Pirzada, S. J. H., & Siddiqui, A. (2013). Analysis of edge detection algorithms for feature extraction in satellite images. In *Proceedings of the IEEE international conference on space science and communication (IconSpace)* (pp. 238–242). IEEE.

Pranata, Y. D., Wang, K. C., Wang, J. C., Idram, I., Lai, J., Liu, J. W., et al. (2019). Deep learning and SURF for automated classification and detection of calcaneus fractures in CT images. *Computer Methods and Programs in Biomedicine, 171*, 27–37.

Rezai-Rad, G., & Aghababaie, M. (2006). Comparison of SUSAN and sobel edge detection in MRI images for feature extraction. In *Proceedings of the 2nd international conference on information & communication technologies* (pp. 1103–1107). IEEE.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference* (pp. 234–241). Springer. October 5-9, 2015, Proceedings, Part III 1.

Russo, F. (1998). Edge detection in noisy images using fuzzy reasoning. *IEEE Transactions on Instrumentation and Measurement, 47*(5), 1102–1105. Oct.

Sarkar, S., Venugopalan, V., Reddy, K., Ryde, J., Jaitly, N., & Giering, M. (2017). Deep learning for automated occlusion edge detection in RGB-D frames. *Journal of Signal Processing Systems, 88*, 205–217.

Shanmugam, K. S., Dickey, F. M., & Green, J. A. (1979). An optimal frequency domain filter for edge detection in digital pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1*(1), 37–49.

Shen, W., Wang, X., Wang, Y., Bai, X., & Zhang, Z. (2015). 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* (pp. 3982–3991).

Shin, M. C., Goldgof, D., & Bowyer, K. W. (1998). An objective comparison methodology of edge detection algorithms using a structure from motion Task. In *Proceedings of the 1998 IEEE CVPR* (pp. 190–195).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017). Inception-v4, inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818–2826).

Tian, B., & Wei, W. (2022). Research overview on edge detection algorithms based on deep learning and image fusion. *Security and Communication Networks*.

Trujillo-Pino, A., Krissian, K., Alemán-Flores, M., & Santana-Cedrés, D. (2013). Accurate subpixel edge location based on partial area effect. *Image and Vision Computing, 31*(1), 72–90.

Verma, O. P., & Parihar, A. S. (2017). An optimal fuzzy system for edge detection in color images using bacterial foraging algorithm. *IEEE Transactions on Fuzzy Systems, 25*(1), 114–127. Feb.

Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., et al. (2018). Understanding convolution for semantic segmentation. In *Proceedings of the IEEE winter conference on applications of computer vision (WACV)* (pp. 1451–1460). IEEE.

Wang, R. (2016). Edge detection using convolutional neural network. *Advances in neural networks–ISNN 2016: 13th international symposium on neural networks, ISNN 2016* (pp. 12–20). St. Petersburg, Russia: Springer. July 6-8, 2016, Proceedings 13.

Wang, X. (2007). Laplacian operator-based edge detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(5), 886–890.

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

Xu, G. S. (2009). Sub-pixel edge detection based on curve fitting. In *Proceedings of the S international conference on information and computing science* (pp. 373–375). IEEE.

Xuan, L., & Hong, Z. (2017). An improved CANNY edge detection algorithm. In *Proceedings of the 8th IEEE international conference software engineering service science (ICSESS)* (pp. 275–278). Nov.

Yang, K. X., & Sheu, M. H. (2016). Edge-based moving object tracking algorithm for an embedded system. In *Proceedings of the IEEE Asia pacific conference on circuits and systems (APCCAS)* (pp. 153–155), 25-28 Octobe.

Yang, Y., Kou, K. I., & Zou, C. (2016). Edge detection methods based on modified differential phase congruency of monogenic signal. *Multidimensional Systems and Signal Processing, 29*(1), 339–359.

Yildiz, A. R., Öztürk, N., Kaya, N., & Öztürk, F. (2003). Integrated optimal topology design and shape optimization using neural networks. *Structural and Multidisciplinary Optimization, 25*, 251–260.

Yu, Z., Feng, C., Liu, M. Y., & Ramalingam, S. (2017). Casenet: Deep category-aware semantic edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5964–5973).

Zhang, H., Du, Y., Ning, S., Zhang, Y., Yang, S., & Du, C. (2017). Pedestrian detection method based on faster R-CNN. In *Proceedings of the 13th international conference on computational intelligence and security (CIS)* (pp. 427–430). IEEE.

Zhang, K., Zhang, Y., Wang, P., Tian, Y., & Yang, J. (2018). An improved Sobel edge algorithm and FPGA implementation. In *Proceedings of the 8th international congress information communication technology* (pp. 243–248).

Zhao, F., Zhang, W., Yan, Z., Yu, H., & Diao, W. (2019). Multi-feature map pyramid fusion deep network for semantic segmentation on remote sensing data. *Journal of Electronics & Information Technology, 41*(10), 2525–2531.

Zhu, Q. (1996a). Efficient evaluations of edge connectivity and width uniformity. *Image and Vision Computing, 14*(1), 21–34.

Zhu, Q. (1996b). Efficient evaluations of edge connectivity and width uniformity. *Image and Vision Computing, 14*(1), 21–34.

Ziou, D., & Tabbone, S. (1998). Edge detection techniques-an overview. *Pattern Recognition and Image Analysis, 8*, 537–559.