International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Boundary Regularization of Urban Buildings using Mask Region Convolutional Neural Networks

S. Vasavi [1*], P. V. Sai Krishna [2], P. D. L. Nikhita Sri [3]

Submitted: 11/01/2024 **Revised:** 17/02/2024 **Accepted:** 25/02/2024

Abstract: Knowing the shape patterns can be highly helpful for building extraction in very high-resolution satellite (VHRS) images, which is among the most fascinating and difficult topics. For a variety of purposes, including land-cover mapping, managing urban resources, keeping track of natural disasters, and locating illegal structures. Deep-learning-based semantic segmentation networks have significantly improved building footprint generation performance when compared to more traditional processes. Some of the existing methods like CLP-CNN, and RegGAN have disadvantages that are unavoidably impacted by a number of circumstances, such as unpredictable backgrounds, blurry buildings in the background, obstructions, etc. These methods can utilize local texture and context data, but they are unable to record patterns of building shapes. A solution based on deep learning is suggested to regularize building borders in order to overcome this problem. The Mask R-CNN technique is employed for detection and Masking. For boundary regularization of buildings Guided filter is used to refine the output masks. These refined masks of two images at different timelines are used to perform change detection. This method has given the regularized building footprints as the output in the form of masks with the changes detected. The performance of the proposed system is measured and obtained an accuracy of 0.92, F1-score of 0.95, precision score of 0.93 and recall score of 0.98. By incorporating guided filters for boundary regularization and change detection, the proposed method achieves high precision in detecting building changes and offers a versatile, non-intrusive solution for various applications, including land-cover mapping and urban resource management.

Keywords: *Convolutional Neural Network, Mask R-CNN, High-Resolution Satellite Imagery, Boundary Regularization, Semantic Segmentation*

1. Introduction

In order to capture small objects and features like particular buildings, trees, roadways, and other man-made structures, VHRS have a high spatial resolution, usually less than one metre per pixel. These images offer an excellent representation of the earth's surface. Semantic segmentation can be applied on these images in order to regularize Building boundaries. After the segmentation and regularizing the boundaries of two same images at different timelines, change detection can be performed. Such work helps real estate industry, flood management and updation of GIS maps. The building boundary regularization model is built using the satellite images of Mumbai city belonging to Maharashtra of India and a sample specimen is shown in Figure 1. Mumbai is a popular choice for urban studies and research due to its status as a megacity with a complex urban landscape, data accessibility, and policy implications. The proposed model seeks to address challenges including complex urban environments with high population density, unpredictable backgrounds, and dynamic changes.

*[*1]Computer science and engineering department, Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India. E-mail: vasavi.movva@gmail.com*
[2] Computer science and engineering department, Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India
[3] Computer science and engineering department, Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India

Fig 1: Dataset specimen

1.1 Convolutional Neural Networks (CNN):

Directly from pixel images, a Convolutional neural network is created to precisely identify visual patterns. A Convolutional layer, a pooling layer, and a fully connected layer are the layers that make up the model network. The model applies a series of filters on the input image in the Convolutional layer. Figure 2 presents CNN architecture is shown in a schematic diagram.

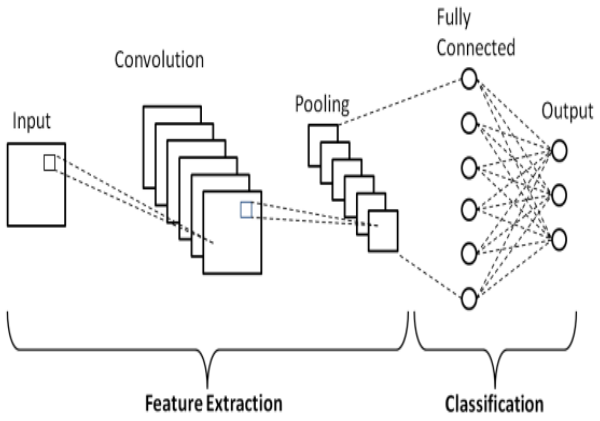


Figure 2: CNN architecture is shown in a schematic diagram.[1]

1.2 Mask RCNN:

The Mask R-CNN extends the Faster R-CNN model by including a segmentation branch, in order to build pixel-level masks for each item recognized in an image. Three key components the mask head, backbone network, and region proposal network (RPN) of the Mask R-CNN architecture are used. In order to gather characteristics from the original image, the backbone network is commonly a pre-trained CNN like ResNet or VGG. Candidate object bounding boxes are proposed by the RPN and subsequently refined using a bounding box regression head. The mask head then applies a series of Convolutional layers to the feature maps in order to create a binary mask for each proposed object. Figure 3 represents its architecture.

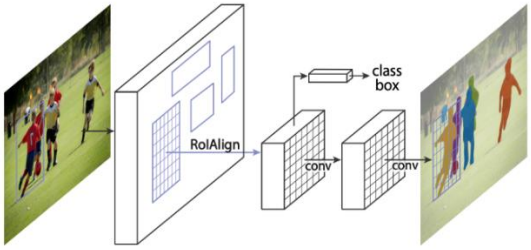


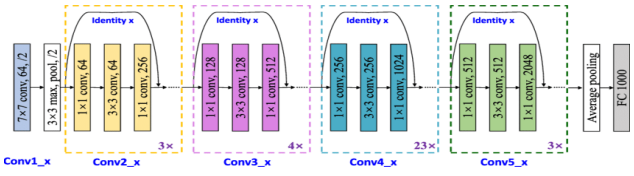
Figure 3: Mask R-CNN Architecture. [2]

1.3 VHRS Images:

Very High-Resolution Satellite (VHRS) images are satellite images with a high spatial resolution, allowing them to capture extremely tiny features on the surface of the Earth. Typically, satellites in low Earth orbit like WorldView-3, Pleiades, and GeoEye are used to take these images.

1.4 Resnet Architecture:

ResNet is known for being able to train neural networks with hundreds of layers or more without experiencing the issue of vanishing gradients that can arise in deep networks. ResNet is made up of a set of residual blocks, each of which has a number of Convolutional layers and shortcut connections. Figure 4 presents the architecture of Resnet-101.



Figur 4: The structure of the ResNet-101-based deep feature extractor. [3]

1.5 Instance Segmentation:

Due to the fact that it calls for both object recognition and semantic segmentation, instance segmentation is a difficult task. The algorithms need to identify the objects in an image and then segment each object into its individual parts. The output of instance segmentation is a set of bounding boxes, one for each object found, and a mask identifying the pixels that make up the object.

1.6 Raster to Vector Data:

Converting raster to vector data refers to image transformation as a process or raster representation into a vector format. Raster data consists of a grid of pixels, where each pixel contains information about color or intensity.

1.7 Shape File:

A common geographical vector data format for storing geographic characteristics like points, paths, and polygons is called a shape file. It consists of multiple files (.shp, .shx, .dbf) that collectively keep track of the features' characteristics and shape.

1.8 Motivation:

Traditionally, boundary regularization of buildings have been carried out through manual mapping, a time-consuming process, labor-intensive, and having more chance of raising errors. There are some disadvantages of using traditional processes and those are unable to solve the problems like land-cover mapping, managing urban resources, monitoring natural disasters and identifying illegal constructions.

The motivation behind this work is to increase the speed and accuracy of mapping, allowing for more efficient and effective method of boundary regularization of buildings by CNN one of the deep learning techniques.

1.9 PROBLEM STATEMENT:

The goal of the project is to regularise the boundaries of buildings in metropolitan areas by creating a model utilising deep learning techniques. In order to capture small objects and features like individual buildings, trees, roads, and other man-made structures, VHRS images must have a high spatial resolution, typically less than one metre per pixel. In order to regularise Building borders and detect changes between two images taken at various timelines, we must employ semantic segmentation from these images. Such work supports the real estate sector, flood management, map updates, and loss estimation during natural disasters.

1.10 Objectives:

The following is a list of this work's main goals:

- To Create Indian dataset specific to the Maharashtra State, Mumbai City.
- To develop a deep learning model that accurately identifies building footprints and regularizes boundaries from VHRS images.
- To update the GIS maps with the change detection

1.11 Organization

This paper follows the following structure: A literature review of various building regularization and change detection efforts is given in Section 2. The proposed approach and architecture are then presented in the section 3. Results achieved and conclusions are discussed in section 4.

2. Related Works

This section provides a list of numerous literature reviews that are used as references. The authors of article [4] described FER-CNN approach as a way to improve building detection and classification accuracy. There are two steps to Faster R-CNN's work. The feature extractor and the VGG16 model are used in the first stage to process the initial data. A map of the features created as a consequence of this method is used in the following step. The second network does structure detection using the regions created by the first network, also known as the (RPN), also referred to as region proposals.

Advantages: FER-CNN method has a substantially more accuracy in the recognition of minute objects (like a garage). Improved resistance to shadow areas, which are frequently visible in satellite images of urban areas, is another advantage of this network.

Disadvantages: Their algorithm does not pick up on objects that are completely shaded and have roofs that only slightly contrast with their surroundings.

Authors of [5] suggested a network for generating building footprints with regularized boundaries and that is named as RegGAN. Based on their nature, tasks can be split into two categories: building boundaries regularization and semantic segmentation. They suggest integrating boundary regularization and semantic segmentation without sacrificing their semantic accuracy. Generator and discriminator are the two modules that make up RegGAN. The regularization path and the reconstruction path are the two main paths of the generator.

Advantages: The best results were obtained using this strategy, which combined boundary regularization with semantic segmentation into an end-to-end network.

Disadvantages: RegGAN requires the use of loss after regularization and discriminator of multiscale to further enhance the regularization of building result.

The method outlined in [6] models the patterns of building shape that improve building segmentation accuracy. An adversarial shape learning network (ASLNet) is suggested by the authors. In ASLNet, the adversarial learning method and a Convolutional Neural Network shape regularizer are both added to explicitly reflect the form restrictions and enhance the embedding of the shape features.

Advantages: It is crucial for many applications, including mapping the land cover, managing resources of urban areas, finding unauthorized structures, etc.

Disadvantages: The ED-FCN produces segmentation results that are typically round-edged.

The LSI-RNN aims to detect line segments directly rather than edges. Its explanation can be found in [7]. In order to produce an (AFM), LSI-RNN employs a neural discriminative dimensionality reduction (NDDR) layer, another contained branch. These method's four consecutive parts are the backbone, NDDRCNNs, AFM, recurrent decoder, and skip features with care. Convolutional features were learned from a given image using the Resnet-50 as a foundation. RNN receives both the location of the initial corner and the skip feature paired with AFM in order to make future predictions of building corners.

Advantages: In three datasets, the LSI-RNN performed better than competing methods and produced the best results. The geometry of the buildings can be described more precisely using this way.

Disadvantages: The LSI-RNN designation has a tendency to highlight the rough edges of structures in order to depict their angular shapes, neglecting the curve's smooth vertices in the process.

According to the methodology described in [8], CLP-CNN technique is used to make the segmentation of boundaries of building automatic from remote sensing images. Three components make up the system:

1) A border predictor is used to gather the rough polygonal bounds of each RoI;

2) To customize the vertices of the retrieved polygonal boundaries, a concentric loop-shaped Convolutional model with bidirectional coupling loss is utilized;

3) At the manual delineation stage, their borders are eventually regularized to polygons using a refinement block.

Advantages: This technique makes it possible to accurately delineate building footprints in surveying and mapping without the need for manual labor.

Disadvantages: It is unavoidably impacted by a number of circumstances, such as unpredictable backgrounds, blurry buildings in the background, obstructions, etc.

Methodology of [9] uses WSSS method, which comprises building CAMs and then training a network for segmentation using those CAMs The methodology contains two networks: BEN and PANet. For PANet, a model that has been pretrained on the City Scapes dataset serves as the foundation for feature extraction.

Advantages: The PANet intends to maximize the quality of CAMs by utilizing global pixel-wise affinities.

Disadvantages: It is reliable to reliable pixel wise affinities only. It is more concentrated on weak supervised buildings.

There are two primary steps in the method described in [10]. The first is a HED unit, which is modeled after the fundamental edge detector, HED, and is coupled to a backbone encoder.

Advantages: It improves automated systems' capacity to recognize the limits of semantic segmentation masks while extracting buildings from remote sensing data.

Disadvantages: The model is implemented in building boundary extraction with U-Net architectures only.

The methodology described in [11] describes a quick border identification method based on a reduction dimensionally strategy in order to increase automation and guarantee correctness. They improved the algorithm upon sleeve, which necessitates a large number of iterations, by converting the borders detected by the alpha shape algorithm to a 2Dimensional image and applying Gaussian filtering recursively. This method increases the accuracy of detection.

Advantages: The technique increases the accuracy of building boundary detection while simultaneously speeding up data processing.

Disadvantages: The overall method generates more accurate building footprints, however it does not take into account the intensity image for further improving the findings through regularization.

In very high resolution aerial images, the methodology in [12] proposed a change detection model based on Convolutional neural networks that can identify instances of changed buildings as well as changing building pixels. These two binary maps are then provided to the change detection network as input. It produced a change map from the two input images of buildings.

Advantages: The framework's unique asset is its ability for self-training, which is crucial for deep learning-based change detection.

Disadvantages: The model is less accurate for the object level segmentation.

In order to extract spatial and spectral correlations as well as additional discriminative properties, the methodology in [13] uses a dual-branch deep network with a parallel spatial-channel attention mechanism. The difference between altered objects and

backdrops was made easier by using spatial attention in deep features, which quantified the rich context of local features.
Advantages: The outcomes showed that this strategy improved the evaluation metrics for accuracy in the two datasets for LEVIR-CD and BCDD.
Disadvantages: As it is particularly difficult to identify the exact shape of building modifications, edges are not removed accurately in locations with high building density.
There are two steps in the process in [14]. Utilizing the bi-temporal images as input is the initial stage. Then, each building's related temporal image was compared using comparison metrics including PSNR and MSE. They used each image's HSV representation for this comparison. SVM and Random Forest were used as two classifiers to determine whether there had been alterations to buildings.
Advantages: This approach only looked at the identification of changes in buildings, neglecting other kinds of objects like tracks and roadways.
Disadvantages: Building detection directly influences change detection, hence increasing the accuracy of this phase is required to increase the accuracy of the whole model.

2.1 Software Requirements
The basic software requirements include**:**
Operating System: Windows 10 or above.
Development Environment: Google Colab.
Python Libraries: numpy,pandas , PIL
Deep Learning Frameworks:Tensor Flow or PyTorch

2.2 Hardware Requirements
The criteria for the functional platform are
RAM: 8GB
Processor: 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz

3. Methods

This section describes the system's design as well as the methodology and dataset that will be applied.

3.1 Architecture:
Figure 5 presents the proposed system model. For prediction and segmentation of the buildings and their boundaries, the model is trained using MASK R-CNN by taking training images and their annotated images.
To predict and segment the building boundaries the images are first preprocessed removing noise and enhancing the image. Training will be done in the back end which contains the backbone network, RPN, and mask head.

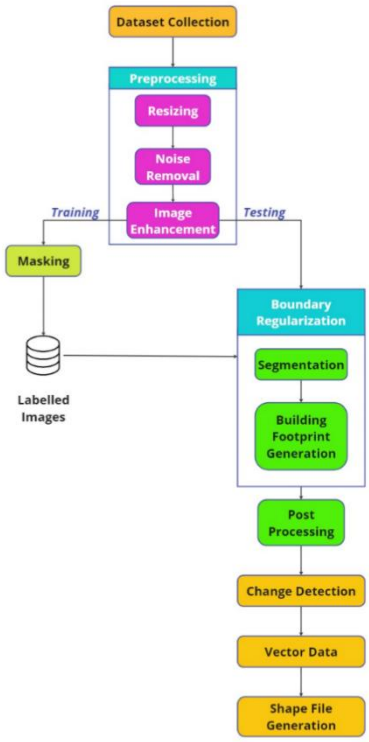


Figure 5: Proposed System Model

Table 1 presents the comparison of the existing Mask R-CNN with the proposed model used in this study.

TABLE1. MODIFIED MASK R-CNN MODEL

Parameters	**Existing Model[25]**	**Modified Model**
Channels in the input image	3 (RGB)	3 (RGB)
The shape of input image	Variable	1024 x1024
Strides	[4, 8, 16, 32]	[4, 8, 16, 32]
Input Kernel size	7x7	7x7
Initial Number of Filters	256	256
Parameters	~30.2 million	~30.2 million
Pooling type	ROI Pooling	ROI Pooling
Size of Max Pooling at every layer	2x2	3x3
Number of Layers	Varies	3
Channels in the output image	81 (including background class)	81 (including background class)

3.2 Methodology:
This section talks about the many sections used in the work. Preprocessing, labeling the images, boundary regularization of buildings using Mask-RCNN, post-processing using Edge Detection, Change Detection using pixel-based change detection

approach, the conversion of the image data into vector data, and the generation of Shape files using vector data for GIS map updating purposes are among the modules included.

3.2.1 Preprocessing:

In this phase, the Preprocessing step involves the following steps:
(i) Resizing:The images are resized from 8192 * 4902 to 1024 *1024.
(ii) Noise Removal: Image noise is removed from the images using the Median filtering technique.And then the PSNR value is calculated which denotes the amount of noise in the before and after images.
(iii) Image Enhancement:The images from which noise is removed then are enhanced using the Unsharp Masking technique.

3.2.2 Labelling

In this phase, the preprocessed images are labeled and they are saved in a folder where training images are saved. These labeled images are utilized to train the model as ground truth images that correlate to the training images. And also for the evaluation of the model the predicted image by the model and the ground truth image are compared.

3.2.3 Boundary Regularization

The Boundary regularization of buildings is carried out using the Mask R-CNN model. It is mainly built using three components:

- Backbone network: ResNet, a CNN, is commonly utilized as the backbone network to extract features from the initial image.
- The Region Proposal Network (RPN) is in charge of recommending potential object regions in the image. Based on the features retrieved from the backbone network, it produces a set of candidate regions and the corresponding scores.
- Mask head: The mask head is responsible for generating binary masks for each detected object instance. It takes as input the region of interest (RoI) proposals generated by the RPN and the features gathered from the backbone network It outputs a binary mask for each detected object instance.

3.2.4 Post-processing:

In this phase-guided filter, a post-processing technique is used to refine the output mask and reduce noise and artifacts. Post-processing is typically applied to a source image to produce a filtered output image that preserves the structure and details of the original image while reducing noise and artifacts.

3.2.5 Change Detection:

In this step a pixel-based change detection approach is discussed. It analyses corresponding pixels in the before and after images and looks for differences in pixel intensity to identify changes. To find the modified pixels, it computes the absolute difference between the binary masks of the before and after images. The system then performs additional analysis on the altered pixels to identify the type of change (such as new construction, demolished structures, or unmodified buildings) based on the pixels in various categories.

3.2.6 Vector Data:

To represent geographical properties or objects with geometric shapes like points, lines, and polygons, vector data is utilized to create a shape file. Vector data is used to provide the borders and characteristics of the features to be represented when creating a shape file. The shape file can be created by specifying the features' geometry and properties using vector data. Geographic Information Systems (GIS) and geographical analytic jobs frequently employ the shape file format because it makes it possible to store and organize spatial data effectively.

3.2.7 Shape File:

A common geospatial vector data format used in GIS software is called a shape file. It is made up of several files that collectively represent attribute and spatial data. Numerous forms of geometric shapes, including points, lines, and polygons, can be stored in the shape file format, which is widely supported.Due to their ease of use, interoperability, and support in a variety of GIS tools, shapefiles are frequently used for storing and transferring spatial data. They can be used for spatial searches, spatial analysis, and integration with other geospatial datasets and tools. They can also be used for visualizing and analyzing geographic aspects.

3.3 Algorithm:

Following is a description of the image preparation algorithm:
Step 1: Noise Removal using Median Filter for each pixel in the image:
Step 1.1: Create a window centered on the pixel with size n x n
Step 1.2: Sort the pixel values in the window in ascending order.
Step 1.3: The median of the sorted pixel values should be used in place of the pixel value.
$O(i, j) = \text{median}(I(x, y))$ (1)
Where n is the window size, which can be adjusted to achieve distinct levels of noise removal
Step 2: Image Enhancement using Guided Filter
Step 2.1: Convert the input image to grayscale.
Step 2.2: Create a blurred version of the grayscale image using a Gaussian filter.
Step 2.3: To create a high-pass filtered image, subtract the blurred image from the grayscale image.
Step 2.4: Multiply the high-pass filtered image by a scaling factor.
Step 2.5: To create the sharpened image, combine the original grayscale image with the scaled high-pass filtered image.
Output image = Original image + Amount of edge enhancement * (Original image - Blurred image) (2)
Where ”Original image” is the input image. ”Blurred image” is the input image convolved with a Gaussian filter to blur it. ”Amount of edge enhancement” is a scalar factor that controls the strength of the edge enhancement. It is usually set to a value between 0 and 1. A higher value results in a stronger edge enhancement.

3.3.2 Labeling Images

In this, a human annotator manually assigns labels to each image by viewing and analyzing the content of the image.
Gather and prepare the images for labeling. This may involve resizing, cropping, or enhancing the images to make them suitable for labelling.

3.3.3 Boundary Regularization of Buildings

Instance segmentation in Mask R-CNN entails identifying the objects in an image and segmenting every instance of the object class separately to provide precise and non-overlapping segmentations for every instance of the object class. This is accomplished by creating a binary mask for each proposal and then improving the masks.
Step 1: Enter an RGB image of a city with buildings.
Step 2: Use the Resnet-101 (CNN) as a backbone network to extract Low-Level Features, Textural Features, Shape Features, and Contextual Features from the input Image.
Step 3: Generate a collection of candidate object regions known as RoI proposals using the backbone network's properties. For each

proposal, the RPN forecasts the bounding box positions and objectness ratings.
Step 4: Align each RoI proposal's features to a set size to ensure precise pixel-to-pixel mapping for ensuing procedures.
Step 5: Determine the item category by running the aligned RoI features through a classifier (usually a fully connected network). For each RoI proposal, class probabilities are generated in this stage.
Step 6: Carry out bounding box regression to improve the object bounding box coordinates.
Step 7: Include a parallel branch for pixel-level segmentation in the network. Each object proposal receives a binary mask prediction based on the RoI characteristics, indicating the segmentation of the object at the pixel level.
Step 8: End-to-end train the network using annotated data, employing object class labels, bounding box coordinates, and instance masks as the ground truth.
Step 9: The network predicts object classes, bounding boxes, and instance masks for new images during the inference phase.
Step 10: Use a guided filter as a post-processing approach to smear the edges of the identified mask.

3.3.4 Post-Processing the image

Perform the post-processing method on regularized building masks using Guided Filter.
The steps involved in applying the guided filter to boundary regularization of buildings using Mask R-CNN are as follows:
Step 1: Generate initial object masks using Mask R-CNN on the input image.
Step 2: Convert the initial object masks to grayscale format.
Grayscale image = R*0.299+ G*0.587+B*0.114 (3)
where R, G, and B are the red, green, and blue channels of the input image.
Step 3: Generate a guidance image using the Canny edge detection algorithm [24]
Step 3.1: Apply Gaussian smoothing to reduce noise in the image:
Smoothed image: S(x, y) = G(x, y) * H(x, y), (4)
where H(x, y) is the Gaussian filter.
Gaussian filter formula: H(x, y) = (1 / (2 * pi * sigma^2)) * exp(-(x^2 + y^2) / (2 * sigma^2)), (5)
where sigma is the standard deviation of the Gaussian kernel.
Step 3.2: Calculate the gradient magnitude and direction at each pixel using gradient operators:
Gradient magnitude: M(x, y) = sqrt(Gx(x, y)^2 + Gy(x, y)^2) (6)
where Gx(x, y) and Gy(x, y) are the gradients in the x and y directions, respectively.
Gradient direction: D(x, y) = atan2(Gy(x, y), Gx(x, y)) (7) where atan2 calculates the arctangent of the ratio of Gy to Gx, taking the sign of both into account.
Step 3.3: Perform non-maximum suppression to keep only the local maximum pixels along the edge direction:
Non-maximum suppressed image: N(x, y) = 0 if M(x, y) is not the maximum in the direction D(x, y), else N(x, y) = M(x, y). (8)
Step 3.4: Apply double thresholding to classify pixels as strong, weak, or non-edge pixels:
Strong edge pixel: P(x, y) = 1 if N(x, y) is above a high threshold.
Weak edge pixel: Q(x, y) = 1 if N(x, y) is between the high and low thresholds.
Non-edge pixel: R(x, y) = 0 otherwise.
Step 3.5: Perform edge tracking by hysteresis to connect weak edge pixels to strong edge pixels in the neighborhood:
Edge map: E(x, y) = 1 if P(x, y) == 1 or (Q(x, y) == 1 and at least one of the eight neighboring pixels is a strong edge.
Step 3.5: Normalize the edge map to obtain values between 0 and 1:
Normalized edge map: G'(x, y) = E(x, y) / max(E(x, y)) (9)
Optional: Further enhancement or refinement of the guidance image based on specific requirements or domain knowledge.
Output: The generated guidance image, denoted as G'(x, y).
Step 4: Apply a guided filter to the grayscale object masks using the guidance image as the guide, with a suitable radius and threshold value.
Step 4.1: Pre-compute mean values:
For the input image I: mean_I = boxFilter(I, r) (where boxFilter is a function that computes the mean value in a local window of size (2r+1)x(2r+1))
For the guidance image P: mean_P = boxFilter(P, r)
Step 4.2: Pre-compute correlation values:
For the input image I: corr_I = boxFilter(I * I, r)
For the guidance image P: corr_IP = boxFilter(I * P, r)
Step 4.3: Pre-compute variance values:
var_I = corr_I - mean_I * mean_I
cov_IP = corr_IP - mean_I * mean_P
Step 4.4: Compute the coefficients a and b:
a = cov_IP / (var_I + eps)
b = mean_P - a * mean_I
Compute the mean of the coefficients a and b in a local Step 4.5: Window:
mean_a = boxFilter(a, r)
mean_b = boxFilter(b, r)
Compute the output image Q:
Step 4.6: Q = mean_a * I + mean_b (10)
In the equation (10), 'r' represents the radius of the local window used for filtering, and 'eps' is a small constant added, where guided_filter is the guided filter algorithm, grayscale_image is the grayscale object mask, guidance_image is the guidance image generated in step 3, radius of the filter window, and eps is the threshold value.
Step 5: Use Otsu's Threshold method [25] to produce final binary masks for each object.
Step 5.1: Compute the histogram of the input grayscale image, denoted as H(i), where i ranges from 0 to L-1 (L is the number of levels in gray color, typically 256).
Step 5.2: By dividing each value by the whole amount of pixels in the image, N, normalize the histogram:
P(i) = H(i) / N
Step 5.3: Calculate the cumulative sum of normalized histogram values:
P_cumulative(i) = ∑[j=0 to i] P(j)
Calculate the cumulative mean of the image intensities:
m_cumulative(i) = ∑[j=0 to i] j * P(j)
Initialize variables for maximum between-class variance (var_max) and optimal threshold value (threshold).
Step 5.4: Iterate through all possible threshold values from 0 to L-1:
Compute the background probability: w0 = P_cumulative(t)
Compute the foreground probability: w1 = 1 - w0
Compute the background mean: m0 = m_cumulative(t) / w0 (avoid division by zero)
Compute the foreground mean: m1 = (m_cumulative(L-1) - m_cumulative(t)) / w1 (avoid division by zero)
Compute the between-class variance: var_between = w0 * w1 * (m0 - m1)^2
Update var_max and threshold if var_between is greater than var_max.

Step 5.5: The threshold that maximizes the between-class variation provides the ideal threshold value.
Once you have the optimal threshold value, you can apply it to the original image to obtain the binary image with foreground and background regions separated.
The chosen threshold value for the filtered image is the optimal threshold value that is given by the threshold that maximizes the between-class variance.

3.3.5 Change Detection

Change detection is performed using a pixel-based change detection approach. It involves the following steps:
Step 1: Load the before and after images.
Step 2: Extract the file names without extensions.
Step 3: Obtain the paths to the before and after mask images based on the provided folders and file names.
Step 4: Load the mask images as grayscale.
Step 5: Convert the grayscale mask images to binary images using thresholding.
Step 6: Compute the change mask by taking the absolute difference between the before and after masks and thresholding the result.
Step 7: Perform bitwise operations between the before and after masks to detect changes in buildings:
demolished_buildings represent buildings present in the before mask but not in the after mask.
new_buildings represent buildings present in the after mask but not in the before mask.
unchanged_buildings represent buildings present in both the before and after masks.
Step 8: Count the number of non-zero pixels (white pixels) in each category.
Step 9: Determine the change type based on the counts of demolished and new buildings.
Step 10: Create visualizations by highlighting the changes on the before and after images:
Step 10.1: Mark demolished buildings in red.
Step 10.2: Mark new buildings in green.
Step 10.3: Mark unchanged buildings in blue.
Step 11:Display the visualizations.

3.3.6 Vector Data

The vector data contains information about the spatial location and attributes of the features.
The process of converting raster data into vector data involves several steps. Here is an algorithmic overview of the conversion process:
Step 1: Read the raster image in PNG format
Step 2: Preprocess the image: Perform any necessary preprocessing steps on the raster image to enhance the quality and extract meaningful information. This include operations like resizing, filtering, thresholding and edge detection.
Step 2.1: Resizing: Images are resized from 8192*4902 to 1024*1024.
Step 2.2: Noise Removal(Median Filter):O(i, j) = median(I(x, y)), where (i, j) represents the coordinates of a pixel in the output image, and (x, y) represents the coordinates of the corresponding pixel within the window centered at (i, j).
The median(I(x, y)) function calculates the median value of the pixel intensities within the window at position (x, y) in the input image I.
Step 2.3: Image Enhancement(Unsharp Masking):
Start with the input image, denoted as I(x, y).
Mask(x, y) = I(x, y) - B(x, y)(Blurred image).
AdjustedMask(x, y) = Mask(x, y) * factor.
Sharpened(x, y) = I(x, y) + AdjustedMask(x, y).
Step 3: Identify and extract features: Identify the features or objects of interest in the raster image. This is done by the feature extraction algorithm and canny edge detection algorithm.
Step 4: Convert boundaries to vector format: Convert the extracted boundaries or contours into a vector format representation, such as polygons, polylines, or points. Each boundary represents a separate feature, and it can be represented as a set of coordinates or vertices.
Step 4.1: For each boundary or contour, process the boundary points or pixels.
Step 4.2: If the boundary is already in a pixel-based format, use the contour tracing algorithm and Moore-Neighbor tracing algorithm to extract the sequence of connected points. The Moore-Neighbor tracing algorithm is a simple and efficient algorithm for contour tracing. It follows the boundary pixels in a clockwise direction. The steps are as follows:
Step 4.2.1: Start from a specified point on the boundary.
Step 4.2.2: Check the neighboring pixels in a clockwise manner (starting from the top-left neighbor).
Step 4.2.3: Move to the first neighbor pixel that is part of the boundary (i.e., has a value indicating an edge or boundary).
Step 4.2.4: Continue tracing the boundary by following the boundary pixels in a clockwise direction until you reach the starting point again, forming a closed contour.
The algorithm keeps track of the current position and the direction of the previous step to ensure correct traversal of the contour.
Step 4.3: If the boundary represents a closed loop, apply the polygonization technique, Douglas-Peucker algorithm to convert it into a polygon. The Douglas-Peucker algorithm is a widely used algorithm for polygonization. It simplifies a curve by recursively removing unnecessary vertices while preserving the overall shape. The algorithm works as follows:
Step 4.3.1: Given a curve represented by a sequence of points, find the point with the maximum distance from the line segment connecting the start and end points.
Step 4.3.2: If the maximum distance is greater than a specified threshold, split the curve at that point and recursively apply the algorithm to both sub-curves.
Step 4.3.3: Repeat the process until no points exceed the threshold distance. The resulting points form the polygonal approximation of the curve.
Step 4.4: If the boundary is a curve, apply the curve fitting technique Polynomial fitting to approximate it with polylines or splines. Polynomial fitting involves fitting a polynomial function to the givenpoints. The complexity of the curve depends on the degree of the polynomial.The most common method for polynomial fitting is the least squares method, where the polynomial coefficients are adjusted to minimize thedata points and the curve. Higher degree polynomials can capture more complex curves but may also be prone to overfitting.
Step 4.5: Store the vector representation of the boundary, which can include a set of coordinates or vertices that define the shape.
Step 5: Assign attributes: Assign attributes or properties to the vector features if needed. This can include assigning unique identifiers, labels, or any other relevant attributes to describe the features. The attributes are described as follows:
Unique identifier: Assign a unique identifier to each vector feature. This can be a numerical or alphanumeric value that uniquely identifies the feature within the dataset.
Step 6: Create a vector data structure: Create a data structure, such as a shape file or a GeoJSON file, to store the vector data. Define

the appropriate schema or fields to accommodate the attributes and geometry of the features.
Step 7: Store vector features: Store the vector features in the data structure by adding them as individual features with their corresponding attributes and geometry.
Step 8: Save the vector data: Save the vector data structure, such as the shape file or GeoJSON file, to disk. This allows for easy sharing, visualization, and analysis of the converted vector data.

3.3.7 Shapefile

Step 1: Create a new shape file: Initialize a new shape file object with the desired shape file type (e.g., point, line, or polygon). Specify the shape file path and the shape file mode (e.g., read or write).
Step 1.1: Import the required libraries: Import the shape file library and the geopandas library.
Step 1.2: Initialize a new shape file object: Create a new instance of a shape file object with the desired shape file type (point, line, or polygon). This object will be used to write the geometries and attributes to the shape file.
Step 1.3: Specify the shape file path and mode: Provide the file path and name for the new shape file. You'll also need to specify the mode, which can be "w" for write or "r" for read. If the shape file already exists, you may need to delete it or choose a different file path to avoid overwriting existing data.
Step 1.4: Define the shape file fields: Specify the attribute fields that will be associated with the shape file features. This includes the field names and data types for storing attributes such as unique identifiers, labels, or other relevant information.
Step 1.5: Write geometries and attributes: Use the shape file object to add geometries and their associated attribute data to the shape file. This involves creating feature objects, setting their geometry and attribute values, and appending them to the shape file.
Step 1.6: Close the shape file: After writing all the desired features, make sure to close the shape file to ensure that the data is properly saved and the file is released.
Step 2: Define the shape file attributes: Add attribute fields to the shape file to store additional information associated with the vector features. Define the field names, types, and lengths as needed. Define attribute fields in a shape file using equation (11). This formula and description illustrate the concept of defining attribute fields in a shape file, allowing for the storage of various types of information associated with the vector features.

Attribute Field = Field Name (Field Type, Field Length) (11)

Where, Attribute fields are used to store additional information in a shape file. Each attribute field has a name, a type, and a length to define its properties. The field name is a descriptive name for the attribute. The field type determines the data type of the attribute (e.g., character, numeric, date). The field length specifies the maximum length or size of the attribute value.
Step 3: Open the source vector data: Open the source vector data that contains the features you want to convert into the shape file. The file format is GeoJSON.
Step 4: Iterate over the features: For every feature in the source vector data, extract the geometry and attribute values. Iterate over the features as follows:
Step 4.1: Read the vector data using a suitable library like Geopandas.
Step 4.2: Iterate over the features using a for loop.
Step 4.3: Extract the geometry of the feature.
Step 4.4: Extract the attribute values of the feature.
Step 4.5: Extract the geometry and attributes within the loop.
Step 5: Add features to the shape file: Add each feature as a new shape file record. Set the geometry of the record using the extracted geometry, and assign attribute values to the corresponding fields. To add features to a shape file by creating new shape file records with extracted geometry and attribute values, follow these steps:
Step 5.1: Initialize a new shape file object.
Step 5.2: Define the shape file attributes (fields).
Step 5.3: Iterate over the features.
Step 5.4: Set the geometry of the record using the extracted geometry.
Step 5.5: Assign attribute values to the corresponding fields.

3.4 Dataset Collection:

Very High-Resolution Satellite Images are taken from GoogleEarthPro for the Mumbai city at different timelines in the year 2015 and 2022. From the total 150 images, 120 images are taken as training and remaining images are taken for testing the proposed system. From the dataset provided by National Remote sensing center (NRSC) for the Hyderabad city, images are taken for validation. Figure 6 presents the sample image taken from GoogleEarthPro. Figure 7 represents the study area of Mumbai city. Figure 8 represents the Validation image provided by NRSC.

Fig 6: Mumbai city during the year 2022

Fig 7 Study area

3.5. Evaluation Metrics:

Some of the measurements taken are listed below:

A range of metrics, such as precision (P), recall (R), crossroads upon union (IoU, Jaccard Index), and F1 score (Dice coefficient), are used for evaluating the proposed system's effectiveness.

The number of pixels that were correctly detected as buildings is known as True Positive (TP). The number of pixels that were incorrectly classified as backgrounds is known as false positives (FP). The number of pixels that were correctly classified as backgrounds is known as True Negative (TN). The amount of pixels that were mistakenly classified as structures is known as False Negative (FN)

The IoU ranges between 0 and 1, where a higher value indicates a better overlap between the predicted and ground truth masks. An IoU of 1 means a perfect match, while an IoU of 0 means no overlap at all.

$$IOU = \frac{TP}{(TP+FP+FN)} \quad (12)$$

$$Mea^{\sum_{i=1}^{N} IOU_i}/_{N} \quad (13)$$

Where IOU1, IOU2, ...,IOUn are the IoU values for each class, and N is the total number of classes.

Fig 8 Validation image

The classification metrics are Accuracy, Precision, Recall, and F1 Score.

Accuracy: The classifier's whole precision is measured by accuracy.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (14)$$

Precision: The percentage of accurately anticipated positive samples out of all expected positive samples is known as precision.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (15)$$

The percentage of accurately anticipated positive samples among all actual positive samples is calculated as recall.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (16)$$

The harmonic mean of recall and precision, which is the F1 score, offers a fair comparison of the two.

$$\text{F1 score} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}} \quad (17)$$

Where TP: True Positive (number of correctly predicted positive samples)

TN: True Negative (number of correctly predicted negative samples)

FP: False Positive (number of incorrectly predicted positive samples)

FN: False Negative (number of incorrectly predicted negative samples)

FPR: False Positive Rate (FPR) is a metric represents the proportion of actual negative instances (other than buildings) that are incorrectly classified as positive instances (buildings).

The false positive rate is a statistical concept that measures the probability of a test or diagnostic procedure incorrectly indicating the presence of a condition or attribute when it is not actually present. In other words, it quantifies the likelihood of a false positive result.

Lower false positive rates are generally desirable because they indicate a lower probability of incorrect positive results.

$$FalsePositiveRate = {}^{FP}/(FP + TN) \quad (18)$$

4. Results and Discussions

This section discusses the outcomes produced by the suggested system. Figure 9 to figure 11 presents the sequence of outputs produced in various phases of the proposed system.

Fig 9: Resized Image 8192*4902 to 1024*1024.

Fig 10: After Noise removal using Median filter.

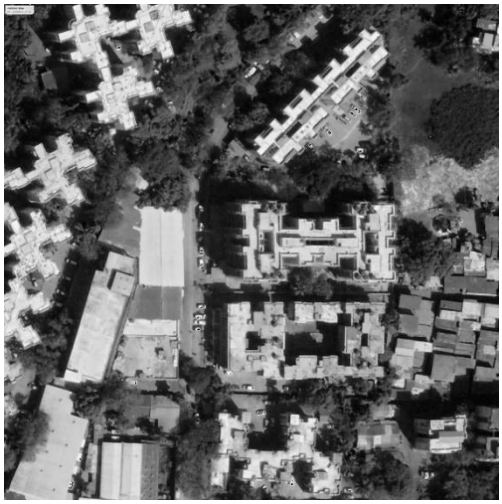


Fig 11: Enhanced Image using unsharp masking.

Fig 12: Image for the year 2022.

Fig 13: Predicted Image of 2022.

Figure 14 presents the original image of 2015 this gives the predicted image to which post-processing is added to get the regularized predicted mask as shown in Figure 15.

Fig 14: Image of 2015.

Fig 15: Predicted Image of 2015.

The images of the same area at two different timelines are taken to detect the changes and classify which buildings are demolished, newly constructed, and unchanged. These classifications are shown in the following colors that are red, green and blue. These predicted images are taken as input for detecting the changes. The detected changes are shown according to the colors as follows: The redcolor represents the demolished buildings, the green color represents changed buildings and blue color represents unchanged buildings and the final image gives the total change detected.

Figure 16, Figure 17, and Figure 18 present the images with red color for demolished buildings, green color for changed buildings, and blue color for unchanged buildings. Figure 19 presents the change detection over two time periods.

Fig 16: Demolished Buildings.

Fig 17: Changed Buildings.

Fig 18: Unchanged Buildings.

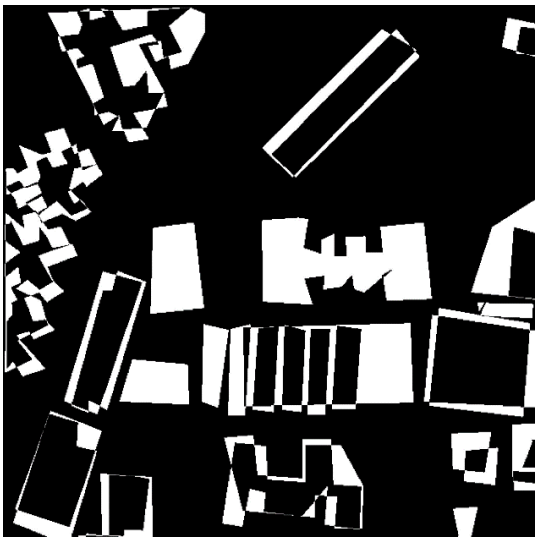


Fig 19: Changed Detected Image.

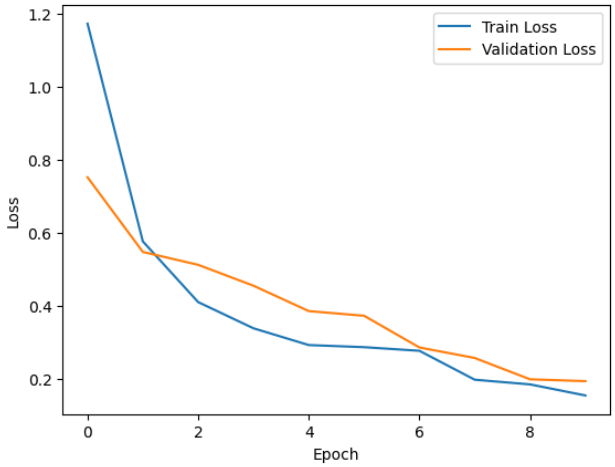


Fig 20: Epoch vs Loss curve

A typical scenario for a well-performing model is that the loss curve decreases rapidly initially and then converges to a lower value as training progresses. In the curve shown in Figure 20, as epochs increase the loss curve is converging to a smaller value which represents the improvement in its predictions.

A graphical representation of the performance of a semantic segmentation or object identification model across various Intersection over Union (IoU) thresholds is the IoU (Intersection over Union) curve and is required for the tasks where evaluating the overlap between anticipated and ground truth regions is required. In Figure21, it can be observed that the curve between the percentage of objects and the IoU threshold typically exhibits an increasing trend. As the IoU threshold becomes more lenient (i.e., lower threshold), the percentage of correctly detected or segmented objects tends to increase. The accuracy may initially be quite poor because the model only has a few training epochs. This is due to the model's early learning stages and the fact that it has not yet learned enough from the data. The model's accuracy tends to rise with the number of epochs. As the model gains knowledge from the training data and modifies its weights and biases to produce more accurate predictions, the curve exhibits an increasing trend. The accuracy might eventually peak or plateau. This shows that the model has learned everything it can from the training data and that adding more epochs has little effect on accuracy. Figure 22 presents Epochs vs Accuracy Curve.

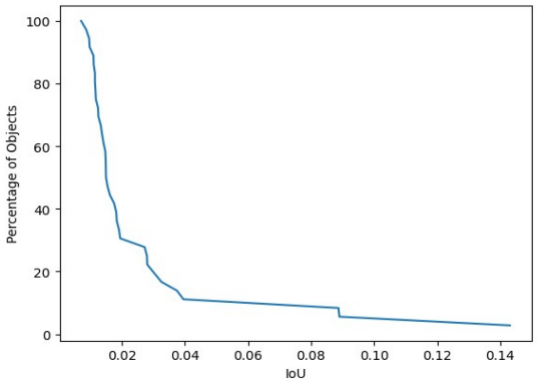


Fig 21: IOU Curve

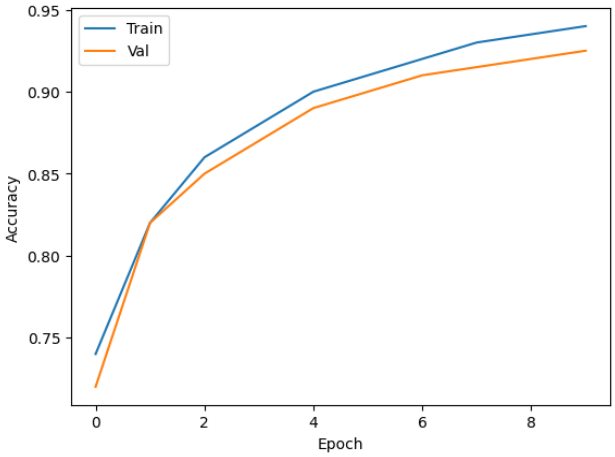


Fig 22: Epochs vs Accuracy

The Performance analysis of the model is measured by calculating precision, f1-score, recall, and accuracy by validating the model using the Mumbai dataset. Accuracy of 92.27% and FPR is 5.24%. Table 2 and Table 3 present the performance of the proposed system during the training, testing, and validation phases. Table 4 compares the performance of the system with the existing approaches.

TABLE 2. Comparison Of Performance In Different Phases

Phase	Precision	Recall	F1-Score	Accuracy
Training	0.95	0.98	0.95	0.93
Testing	0.92	0.96	0.94	0.91
Validation	0.89	0.94	0.91	0.92

TABLE 3. PERFORMANCE METRICS

Metric	Value
True Positive	30.43%
False Positive	3.42%
False Negative	4.31%
True Negative	61.84%
Accuracy	92.27%
False Positive Rate	5.24%

TABLE 4. COMPARISON WITH OTHER MODELS

Work	Dataset	Methodology	Accuracy Test	Accuracy
[6]	Massachusetts Dataset	ASLNet	F1-Score	88%
[16]	Presidente Prudente/Brazil dataset	WICDS	F1-Score	94%
[26]	Inria Dataset	FCN	F1-Score	86%
[4]	Inria Dataset	FER-CNN	F1-Score	85%
[22]	Chandigarh	Ensemble of ResNet and U-Net Model	F1-Score	93%
Proposed Model	Chandigarh	Mask R-CNN	F1-Score	91%

In the proposed model Mask R-CNN is used for the prediction of masks and additionally, post processing technique guided filter is used for more accurate regularized building boundaries and change detection is done by pixel-based change detection model. This study's accuracy metric makes sure the model accurately recognizes building boundaries. The F1-score guarantees that the model effectively minimizes false positives (areas that are mistakenly categorized as buildings when they are not) and false negatives (areas that are incorrectly classified as buildings when they are not), improving the overall quality of the segmentation findings.

5. Conclusion and Future work

For border regularization and distant sensing, CNN-based image processing has been extensively employed. The dataset of Mumbai city is prepared from GoogleEarthPro at 2015 and 2022 timelines. For Preprocessing the images, Noise removal of the image is done using a median filter and the image is enhanced using unsharp masking. For labeling images, the images are labeled manually for training and testing purpose. For Boundary Regularization of Buildings using Mask R-CNN, features are extracted using backbone network i.e. Resnet101, RPN is used for proposing candidate object regions in the image, and Mask head is used to form masks on the object regions. For post processing, guided filter is applied to the masked image for regularizing the building boundaries. Then the images of the same places at different timelines are given to the model to predict the buildings and change is detected between the images. The shape file is also generated and used for the updation of GIS maps using QGIS software. The performance is measured by calculating precision, f1score, recall, and accuracy. This method has obtained an accuracy of 0.92, F1score of 0.91, precision score of 0.89, and recall score of 0.94. Future work includes increasing the accuracy and extending the model to classify the buildings as residential, industry and holy places.

Conflict Of Interest

The authors declare no conflict of interest.

Author Contributions

S.Vasavi: Conceptualization, Methodology, Writing- Original draft preparation, Validation, Reviewing and Editing.
P.V.Sai Krishna and P.D.L.Nikhita Sri: Software, Writing, Original draft preparation, Visualization, Validation.

Acknowledgment

The Authors would like to thank Dr K.Venugopala Rao, Scientist SG (Retd), Group Director Urban studies NRSC, ISRO, Dr SC

Jayanthi, Scientist SG, Group Head, Urban Studies and Applications (USAG), National Remote Sensing Centre (NRSC), Ms J.KAMINI, Sci/Eng. SF HEAD, Urban Studies DIV. NRSC, Ms. Reedhi Shukla, Scientist/Engineer 'SE', Urban Studies Division, NRSC, ISRO, for providing the required data during evaluating the model.

Data Availability Statement

The training and testing data presented in this article are publicly available in [SASPlanet at [http://sasgis.org/]. Validation data utilized in this study were obtained from [NRSC,ISRO]. Data are available from the authors upon request, and with permission from [third party].

References

[1]. Phung, Van Hiep, and Eun Joo Rhee. 2019. "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets" Applied Sciences 9, no. 21: 4500. https://doi.org/10.3390/app9214500

[2]. He, K., Gkioxari, G., Dollár, P., &Girshick, R. (2017)," Mask R-CNN", ArXiv, https://doi.org/10.48550/arXiv.1703.06870

[3]. Jiayao Chen, Mingliang Zhou, Dongming Zhang, Hongwei Huang, Fengshou Zhang,"Quantification of water inflow in rock tunnel faces via convolutional neural network approach",Automation in Construction,Volume 123,2021,103526, https://doi.org/10.1016/j.autcon.2020.103526

[4]. Reda K ,Kedzierski M," Detection, Classification and Boundary Regularization of Buildings in Satellite Imagery Using Faster Edge Region Convolutional Neural Networks", Remote Sens. 2020, 12, 2240. https://doi.org/10.3390/rs12142240

[5]. Li Qingyu, Stefano Zorzi, Yilei Shi, Friedrich Fraundorfer, and Xiao Xiang Zhu," RegGAN: An End-to-End Network for Building Footprint Generation with Boundary Regularization" ,Remote Sensing.2022,14, no. 8: 1835, https://doi.org/10.3390/rs14081835.

[6]. L. Ding, H. Tang, Y. Liu, Y. Shi, X. X. Zhu and L. Bruzzone, "Adversarial Shape Learning for Building Extraction in VHR Remote Sensing Images", in IEEE Transactions on Image Processing, vol. 31, pp. 678-690, 2022, doi: 10.1109/TIP.2021.3134455.

[7]. Z. Liu, H. Tang and W. Huang, "Building Outline Delineation From VHR Remote Sensing Images Using the Convolutional Recurrent Neural Network Embedded With Line Segment Information", in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-13, 2022, Art no. 4705713,doi: 10.1109/TGRS.2022.3154046. .

[8]. S. Wei, T. Zhang and S. Ji,"A Concentric Loop Convolutional Neural Network for Manual Delineation-Level Building Boundary Segmentation From Remote-Sensing Images", in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-11, 2022, Art no. 4407511, doi: 10.1109/TGRS.2021.3126704.

[9]. X. Yan, L. Shen, J. Wang, Y. Wang, Z. Li and Z. Xu,"PANet: Pixelwise Affinity Network for Weakly Supervised Building Extraction From High-Resolution Remote 37 Sensing Images", in IEEE Geoscience and Remote Sensing Letters, vol. 19, pp. 1-5, 2022, Art no. 6515905, doi: 10.1109/LGRS.2022.3205309.

[10]. H. Jung, H. -S. Choi and M. Kang, "Boundary Enhancement Semantic Segmentation for Building Extraction From Remote Sensed Image", in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-12, 2022, Art no. 5215512, doi 10.1109/TGRS.2021.3108781.

[11]. Y chen, Wei Yao, "Extraction of Orthogonal Building Boundary from Airborne Lidar Data Based on Feature Dimension Reduction", DOI:10.5194/isprs-annals-V2-2022-351-2022

[12]. Ji, Shunping, Yanyun Shen, Meng Lu, and Yongjun Zhang. 2019. "Building Instance Change Detection from Large-Scale Aerial Images using Convolutional Neural Networks and Simulated Samples" Remote Sensing 11, no. 11: 1343. https://doi.org/10.3390/rs11111343

[13]. AkramEftekhari, Farhad Samadzadegan, FarzanehDadrassJavan,"Building change detection using the parallel spatial-channel attention block and edge-guided deep network" International Journal of Applied Earth Observation and Geoinformation, Volume 117,2023, 103180,ISSN 1569-8432. https://doi.org/10.1016/j.jag.2023.103180.

[14]. Diego Alonso Javier Quispe and Jose Sulla-Torres, "Automatic Building Change Detection on Aerial Images using Convolutional Neural Networks and Handcrafted Features" International Journal of Advanced Computer Science and Applications(IJACSA), 11(6), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110683

[15]. Luis Rodriguez-Benitez, Carlos C ordoba Ruiz, Luis Caba nero G omez, Ram on Herv as, Luis Jimenez-Linares,An Internet of Agents Architecture for Training and Deploymentof Deep Convolutional Models,Journal of Signal Processing Systems (2022) 94:283– 291,https://doi.org/10.1007/s11265-020-01602-6

[16]. Xu, Yongyang, Liang Wu, Zhong Xie, and Zhanlong Chen. 2018. "Building Extraction in Very High Resolution Remote Sensing Imagery Using Deep Learning and Guided Filters" Remote Sensing 10, no. 1: 144. https://doi.org/10.3390/rs10010144

[17]. K. Wu, W. Dong, Y. Cao, X. Wang and Q. Zhao, "An Improved Method of Median Filtering Forensics for Enhanced Image Security Detection," 2021 International Conference on Networking and Network Applications (NaNA), 2021, pp. 308-312, doi: 10.1109/NaNA53684.2021.00060.

[18]. M. A. Badamchizadeh and A. Aghagolzadeh, "Comparative study of unsharp masking methods for image enhancement," Third International Conference on Image and Graphics (ICIG'04), 2004, pp. 27-30, doi: 10.1109/ICIG.2004.50.

[19]. K. He, G. Gkioxari, P. Doll´ar and R. Girshick, "Mask R-CNN," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 386-397, 2020, doi:10.1109/TPAMI.2018.2844175.

[20]. K. He, J. Sun and X. Tang, "Guided Image Filtering," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 6, pp. 1397-1409, 2013, doi: 10.1109/TPAMI.2012.213.

[21]. Keiron O'Shea1 and Ryan Nash, An Introduction to Convolutional Neural Networks, arXiv:1511.08458, pp:1-11, 2015, https://doi.org/10.48550/arXiv.1511.08458

[22]. A. Mohammad, O. S. Gullapalli, S. Vasavi and S. Jayanthi, "Updating of GIS maps with Change Detection of Buildings using Deep Learning techniques," 2022 International Conference on Futuristic Technologies (INCOFT), 2022, pp. 1-6, doi: 10.1109/INCOFT55651.2022.10094545.

[23]. K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[24]. Canny, J., A Computational Approach To Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[25]. Liu, Dongju (2009). "Otsu method and K-means". Ninth International Conference on Hybrid Intelligent Systems IEEE. 1: 344–349.

[26]. J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965