



(RESEARCH ARTICLE)



## Monitoring urban growth using deep learning-based model for building detection

Reedhi Shukla \*, Sampath Kumar P, Satish Jayanthi and Kamini J

*National Remote Sensing Centre, ISRO, Hyderabad, India.*

International Journal of Science and Research Archive, 2024, 13(02), 1245–1250

Publication history: Received on 24 November 2024; revised on 17 November 2024; accepted on 19 November 2024

Article DOI: <https://doi.org/10.30574/ijrsra.2024.13.2.2142>

### Abstract

Urban growth is an important criterion for understanding the city's development. Buildings are essential to understand the need for new development and growth that happened in recent years. Automatic extraction of building footprints, especially in Indian cities scenario, is of great importance, as it helps in understating the pattern of urban growth, monitoring any illegal construction and change analysis for different time periods. This research will focus on how deep-learning based methodology will help in the automatic extraction of building footprints for multiple Indian cities.

**Keywords:** Deep-learning; building footprints; python; Keras; Tensor flow; U-Net; Satellite; Urban

### 1. Introduction

Detection of urban features in satellite data is always a time consuming and tedious task. Automatic detection of urban features like building plays an important research whenever urban growth monitoring is required. With the easy accessibility and increasing resolution of satellite imagery, more research effort has been recently put on extracting urban features from very high-resolution satellite images [1]. The recent research in deep leaning in branch of computer science has given the way to develop the algorithms which can help in automatic detection of building in satellite data. This paper will provide the deep learning methodology where 2d neural networks is trained using identified samples from satellite data and trained neural network is used for detection of building in very high resolution satellite data.

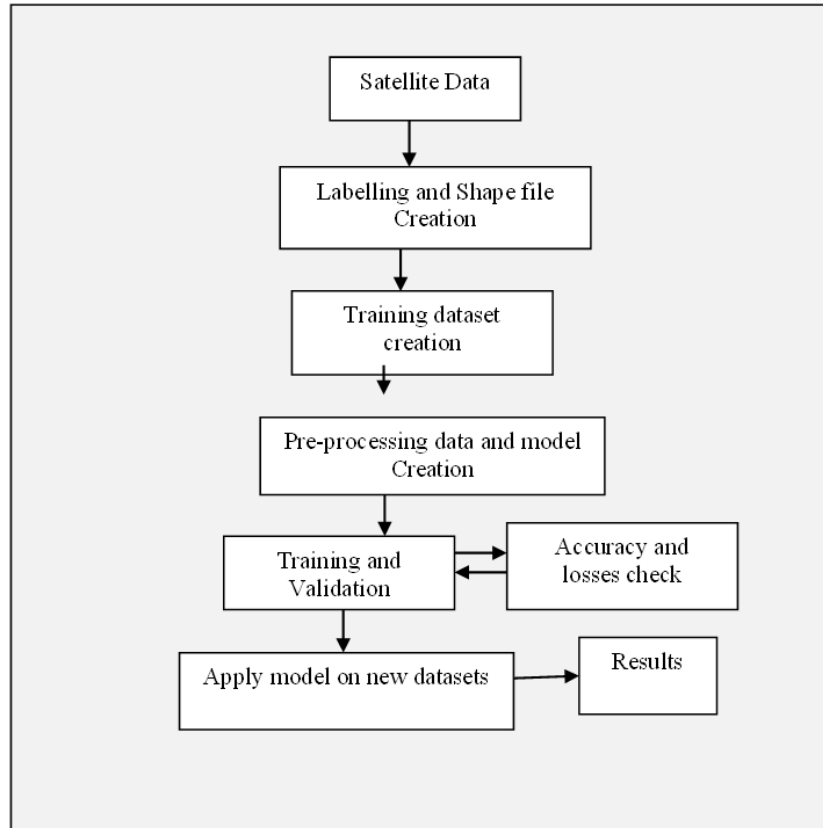
### 2. Literature survey and background study

Before starting of this work we have studied the research in field of object detection in satellite imagery. The Git hub has provided some content which has given the knowledge about the ImageAI which is a python library built to empower developers, researchers and students to build applications and systems with self-contained Deep Learning and Computer Vision capabilities using simple and few lines of code.[2].This ImageAI library was used to detect the objects in the photo graphs. Further exploring about the detection of feature in satellite imagery we got knowledge about the Convolution Neural Networks to detect features in satellite images. CNNs are used for face recognition, object detection, analysis of medical images, automatic inspection in manufacturing processes, natural language processing tasks, and many other applications.[3].Building, evaluating, training and optimizing a neural network required the knowledge of Tensor flow. TensorFlow machine learning framework created by Google is very rich in libraries which can be used for numerical computations.[4].Plane Detection in satellite datasets have better understanding of how to detect features in the satellite data[5]. The above finding helped in developing the methodology for the building detection in the satellite data.

\* Corresponding author: Reedhi Shukla

### 3. Methodology

Here the main goal is data collection, obtaining labels for training, and determining the accuracy value. Data flow diagram of the proposed system is shown in Fig. 1. The dataflow diagram is the form of representation of flow of data in a system. [8]



**Figure 1** Flow diagram of Building detection system

#### 3.1. Data collection

VHRS data set with 1 meter is collected and used for the preparation of training data sets. Fig. 2 is showcasing the sample data sets used for the preparation of the training dataset.



**Figure 2** Satellite data set for predicting the buildings

### 3.2. Labeling and Shape file Creation

The satellite data is visualized in Quantum gis and preparation of shape file is done by marking building and non-building objects in satellite data by visual interpretation and saving as shape file with attribute named as status values as 0 or 1 .If feature is building the attribute value is 1 if feature is non-building the status is 0. The shape file will be saved in the local system for future reference.



Figure 3 Shape file creation and labeling of objects in QGIS

### 3.3. Sample Creation

```

tilesize = 30
feats = vlayer.getFeatures()
fnamelist = [da_in.GetDescription().split('/')]
fnamelist = fnamelist[len(fnamelist)-1].split('.')
band_in = da_in.GetRasterBand(1)
geotransform = da_in.GetGeoTransform()
projection = da_in.GetProjection()
idx = vlayer.Fields().IndexFromName('status')
for feat in feats:
    geom = feat.geometry()
    print ("Feature-ID: %s" % str(feat.id()))
    # show some information about the feature
    x = geom.asPoint()
    lonMin = float(qgis.core.GpsPointXY.toString(x).split(",")[0])
    latMin = float(qgis.core.GpsPointXY.toString(x).split(",")[1])
    sampleStatus = feat.attributes()[idx]
    minX = int((lonMin-geotransform[0])/geotransform[1]) #upper leftX
    minY = int((latMin-geotransform[3])/geotransform[5]) #upper leftY
    #print(minX)
    #print(minY)
    for i in range(0, len(geotransform)):
        # print (geotransform[i])
    minX = minX - tilesize/2
    minY = minY - tilesize/2
    #print(minX)
    #print(minY)
    minOut = geotransform[0]+minX*geotransform[1]
    maxOut = geotransform[3]+minY*geotransform[5]
    geotransform1 = list(geotransform)
    geotransform1[0] = minOut
    geotransform1[3] = maxOut
    print (fnamelist[0])
    driver = gdal.GetDriverByName('OTIF' )
                
```

















 0_4_1_2_2_74.40 25802662_22.7752 276774	 0_4_1_2_2_74.40 26451161_22.7748 267007	 0_4_1_2_2_74.40 27100581_22.7748 667984	 0_4_1_2_2_74.40 29597286_22.7752 541398
 0_4_1_2_2_74.40 49029234_22.7764 521986	 0_4_1_2_2_74.40 53748421_22.7771 492912	 0_4_1_2_2_74.40 54100243_22.7765 354704	 0_4_1_2_2_74.40 54489696_22.7775 625957
 1_1_2_2_74.392 8045579_22.80761 25668	 1_1_2_2_74.391 2454819_22.80654 25361	 1_1_2_2_74.396 8237564_22.80662 59525	 1_1_2_2_74.396 9215549_22.80681 86731
 1_1_2_2_74.417 2002613_22.80635 52684	 1_1_2_2_74.418 9290936_22.80770 74888	 1_1_2_2_74.420 3054196_22.79394 22005	 1_1_2_2_74.420 7829046_22.80200 12637

Figure 4 Creation of samples from the satellite imagery

This steps requires the preparation of multiple chip from the satellite data .The shape file prepared in the step2 is taken as input for the preparation sample chips of 30x30 pixels from satellite data .The shape file attribute is read using OGR



and based on the status, location from satellite data the chip of 30 by 30 is created from satellite imagery and naming convention is kept status\_satellite id\_locationid.

### 3.4. Creation of JSON file

This step requires the preparation of json file which will be used as final input for the training and validation of the model. The 30 by 30 chips prepared in the Step3 are used as input for the preparation of JSON file. GDAL and OGR libraries are used along with python in QGIS software for the preparation chips and JSON file automatically. The JSON file is prepared having values corresponding to data, labels, locations and scene-id. The total of 224 sample are being taken for this methodology out of which 47 are positive samples and others 177 are negative samples

```

8      datalist = []
9      ds_in = gdal.Open(filename)
10     datain = ds_in.GetRasterBand(1).ReadAsArray(0, 0, tilesize, tilesize)
11     datalist = list(datain[0:(tilesize),0])
12     for i in range (1, (tilesize)):
13         datalist.extend(list(datain[0:(tilesize),i]))
14     datain = ds_in.GetRasterBand(2).ReadAsArray(0, 0, tilesize, tilesize)
15     for i in range (0, (tilesize)):
16         datalist.extend(list(datain[0:(tilesize),i]))
17     datain = ds_in.GetRasterBand(3).ReadAsArray(0, 0, tilesize, tilesize)
18     for i in range (0, (tilesize)):
19         datalist.extend(list(datain[0:(tilesize),i]))
20     fnamelist = filename.split('\\')
21     fnamelist = fnamelist[len(fnamelist)-1].split(".tif")
22     fcomponents = fnamelist[0].split('_')
23     longlats = fcomponents[2].split('_')
24     #print (longlats)
25     location = [float(longlats[0]),float(longlats[1])]
26     data.append(datalist)
27     labels.append(int(fcomponents[0]))
28     locations.append(location)
29     scenes.append(fcomponents[1])

```

**Figure 4** Creation of JSON file using GDAL and OGR library with python code in QGIS

### 3.5. Preprocessing of Data and batch creation

The JSON file is used for the training of the model. The tensor flow libraries are being used for the reading of JSON file and training of neural networks and building detection in very high resolution satellite data. JSON file have values corresponding to data, labels, locations and scene-id. The data values of JSON file is converted into array and array is converted to a plain list and transposed and stored in variable X. The labels values of JSON file are converted to matrix and stored as variable Y. The pre-processing of data is required in order to train the neural network. The first step is calculation of mean and standard deviation for entire dataset.

### 3.6. Model Building for building detection

This step included the building of the neural networks. 2D Convolutional Layers constitute Convolutional Neural Networks (CNNs) along with Pooling and fully-connected layers and create the basis of deep learning [6]. The present methodology used the neural network with 3 convolutional layers using tflearn. The first layer has 32 convolutional filters with 3 filters, after this downsizing is being done and there will be two convolutional filters in cascade with 64 convolutional filters with filter size of 3. Further downsizing will provide fully connected network with 512 neurons with activation function followed by 40 percent dropouts. The last layer is fully connected network with 2 neurons and activation function to determine the category of the feature in the image. TFLearn provides a model wrapper 'DNN' that can automatically performs a neural network classifier tasks, such as training, prediction, save/restore, et.[7]. The model is trained for 70 epochs with each batch size of 66. After training of the model the model is getting accuracy of 98% with validation datasets. The model is then tested with the validation set of data set and predicted feature class taken from validation samples was predicted correctly as building and non-building. The trained model is then used for detecting the building on four new satellite data set. The satellite data take as input is of 1m resolution and given as input to model as tiff file and trained model is runs on the satellite to detect the building in the satellite imagery. The trained model will run row wise and form rectangular shape window on the feature where building is being detected and these rectangular windows can be converted to a shape file to be used in GIS environment. The model summary and workflow is shown in Fig5.

## 4. Results and Discussions

The proposed methodology is being used on satellite imagery having 1 m of spatial resolution. The methodology is effective in detecting the buildings on the satellite imagery by drawing the bounding box across the building with the accuracy of 85%. The Fig 6 is showcasing the result of the model.



#### 4.1. Limitations and Advantages

The model is validated on satellite with spatial resolution  $\geq 1$ . The results showcasing by the methodology is providing 85% of accuracy. The model is applied only on multispectral satellite data same may not work for the panchromatic high-resolution imagery.

---

#### Compliance with ethical standards

##### *Disclosure of conflict of interest*

No conflict of interest to be disclosed.

---

#### References

- [1] Y. Xiao ; S.K. Lim ; T.S. Tan ; “Seng Chuan Tay Feature extraction using very high resolution satellite imagery”
- [2] "ImageAI Documentation," [Online]. Available: <https://imageai.readthedocs.io/en/latest/>. [Accessed: November, 2022].
- [3] A. Taspinar, "Using Convolutional Neural Networks to Detect Features in Satellite Images," [Online]. Available: <http://ataspinar.com/2017/12/04/using-convolutional-neural-networks-to-detect-features-in-sattelite-images/>. [Accessed: November, 2022].
- [4] "TensorFlow Tutorial - Deep Learning Using TensorFlow," DataCamp Community, [Online]. Available: <https://www.datacamp.com/community/tutorials/tensorflow-tutorial>. [Accessed: November, 2022].
- [5] E. Frenau, "CNN Plane Detection - Final Project for a Class," Kaggle, [Online]. Available: <https://www.kaggle.com/efreneau/cnn-plane-detection-final-project-for-a-class/>. [Accessed: November, 2022].
- [6] "Convolution Layer - Machine Learning Guru," [Online]. Available: [http://machinelearningguru.com/computer\\_vision/basics/convolution/convolution\\_layer.html](http://machinelearningguru.com/computer_vision/basics/convolution/convolution_layer.html). [Accessed: November, 2022].
- [7] "TFLearn: Deep learning library featuring a higher-level API for TensorFlow," [Online]. Available: <http://tflearn.org/tutorials/quickstart.html>. [Accessed: November, 2022].
- [8] A. Femin and K. S. Biju, "Accurate Detection of Buildings from Satellite Images using CNN," 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ICECCE49384.2020.9179232