ISSN 2395-1621

Prediction of Socio-Economic Status of an area using Satellite Image

[#1]Akshay Mahajan, [#2]Hariom Gugnani, [#3]Aakash Mahajan, [#4]Mukul Bagul,
[#5]Mrs. B. D. Shendkar

[#1234]Student, Department of Computer Engineering,
[#5]Assistant Professor, Department of Computer Engineering,

Sinhgad Institute of Technology and Science, Narhe, Pune, Maharashtra, India.

ABSTRACT

The government is unable to estimate socio-economic status of a remote area and also they are unable to help them. Because government only has their satellite image as a record and they can only see that area through map but through this image they cannot get status about that area. So, considering this satellite image of an area, there is a profound need to detect status of the remote area. In this project, we propose an advanced framework to identify socio-economic status of area through satellite image. We are considering some major factors or attributes like water supply, roof tops, electricity and agriculture field and we are going to train some datasets through CNN technique then input satellite image is compare with train datasets and if there is presence of this factors in input image then we classify status of the area as poor, rich or medium.

Keywords- NumPy, OpenCV, Satellite Image , LandSat 7, Google Earth, CNN, Machine Learning, TensorFlow

ARTICLE INFO

Article History

Received: 1st May 2019

Received in revised form :

1st May 2019

Accepted: 5st May 2019

Published online :

06th May 2019

I. INTRODUCTION

There are so many regions in the world where humans are exist but they have no facilities for their livelihood. They don't even have basic necessity of life like water, food and so on. Some region has lack of only one factor and some regions have lack of all the factors. Some region has water but not electricity while another region has home but not any other necessities. For such type of regions, some organizations are ready to help them with the support of government of that country but due to lack of communication from that region, the organization knows only the location of that region. They don't even know what the basic necessities of that region are?

In that case, the organization can only have the satellite image of the region and they try to determine necessities by observing satellite image. But by only observing that region through satellite image we cannot estimate the presence of the factors on that region. So to solve this kind of problem we are introducing an application to predict socioeconomic status of a region.

The system which we are designing has the ability to identify some major factors which are very basic necessities of a region and they are electricity, water supply, agricultural field. One more factors we are using for estimate status of region is roof top of the house. Roof top is a very essential factor for our system. For prediction of socio-economic status, our system takes satellite image and then this satellite image is compared with our trained model which contains all these major factors present within it and after comparing these factors we get prediction of the status of that satellite image in the form of percentage of presence of factors in the image and by considering this percentages of factors we are predicting socio-economic status.

To achieve required result, application is design through python language and using its libraries. So, to design user friendly desktop application, PyQT library method is used in python language. To pre-process datasets of satellite images, we are using OpenCV library method and through pre-processing of image, we converts our image satellite image into grayscale image, contour image and smoothen image. To design trained model, we are using Convolutional Neural Network (CNN) and to authenticate the user, we are using MySQL database connectivity.

The overview of research paper is that it contain related of literature survey and then it contain methodologies used and then conclusion and references.

II. RELATED WORK

They propose a two-step approach for predicting poverty in rural regions of India from satellite imagery. First, they train a multi-task fully convolutional model to predict three developmental parameters – the main material of the roof, source of lighting and source of drinking water – from satellite imagery. Using only satellite imagery as input, they are able to estimate income and poverty close to the true values collected on the ground by significant manual effort and monetary expense. Their main contribution is a two step approach for poverty prediction. First, they engineer a multi-task fully convolutional model to predict the material of roof, source of lighting and source of drinking water from the satellite imagery of a village. They provide a larger dataset for training. Second, they train a model to predict the income levels using the predicted developmental parameter outputs of the first model. The multi-task fully convolutional model enables them to study the relationships between the features learned for the prediction of different developmental parameters. The results presented by them clearly support the effectiveness of their approach. In this way, their experiments suggest that predicting poverty levels from multiple developmental parameters is more reliable than using a single parameter. Their main focus on only three factors which are the material of roof, source of lighting and source of drinking water present in input satellite image.[1]

Data on infrastructure quality outcomes in developing countries is lacking, and this work explored the use of globally available remote sensing data for predicting such outcomes. Using Afrobarometer survey data, introduced a deep learning approach that demonstrates good predictive ability.Their results demonstrate the proof of concept that satellite imagery can be used to predict infrastructure quality.[2]

Their results show that the current state-of the-art in satellite-based poverty prediction lends itself to predicting relative wealth within a single country where some ground truth data is available, but may struggle with extrapolating across country borders. Using some combination of nightlights and predictions from the proposed models may yield further improvements.[3]

Presents the CNN predictions for urban areas using imagery for either Digital Globe or Planet, using the validation sample. They present R2 estimates that show the correlation between predicted poverty and benchmark poverty as measured in the 2015 Intercensus. The drop in performance is modest but not severe. Poverty estimates for urban areas in Mexico are mapped. Their main focus is on comparing predicted poverty level with actual poverty level.[4]

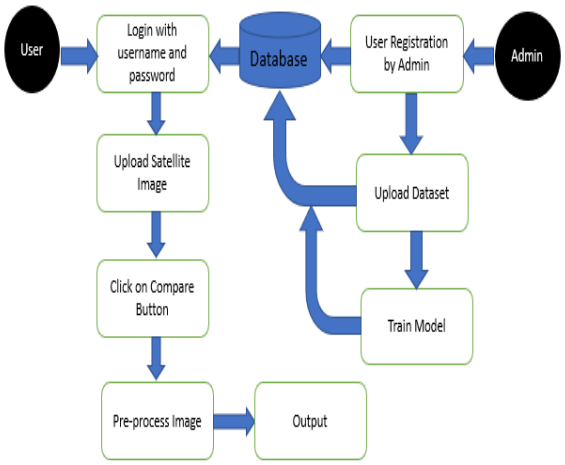

Figure No. 2.1 Proposed System Architecture

In our project, admin handles datasets of all the satellite images and take care of the authentication of the user and another task admin have is to train the model .

User can only access application if they have valid authentication and after successfully authenticate through admin, user can access application and within the application, user has to perform various operation like select input satellite image, select dataset directory, click on preprocess button, operate compare method and so on.

As it is shown in figure that after performing all operations by user and admin, output will be generated in terms of prediction of socio-economic status of a input image.

III. METHODOLOGIES USED

Now it is the time to articulate the research work with ideas gathered in above step by adopting all the below approaches:

A. Open Computer Vision (OpenCV)

A prior knowledge on Python and Numpy is required before starting OpenCV. Python is a general purpose programming language which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal.

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

To install OpenCV in python in linux use command:- pip install opencv.

B. Convolutional Neural Network (CNN)

CNN uses special convolution and pooling operations and performs parameter sharing. To apply CNN method on the

application, there is a need of TensorFlow and Kernos library for the feature extraction. In our application, we are using three layers of CNN.

1. *TensorFlow*

TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. TensorFlow is a framework created by Google for creating Deep Learning models. TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework. TensorFlow is also called a "Google" product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

The steps, which require the execution and proper dimension of the entire network:

1. Include the necessary modules for TensorFlow and the data set modules, which are needed to compute the CNN model.
2. Declare a function called run_cnn(), which includes various parameters and optimization variables with declaration of data placeholders. These optimization variables will declare the training pattern.
3. In this step, we will declare the training data placeholders with input parameters. We can reshape the tensor according to our requirements.
4. Now it's time to create convolutional layers.
5. Let us flatten the output ready for the fully connected output stage - after two layers of stride 2 pooling with the dimensions of 32 x 32.
6. We should set up recording variables. This adds up a summary to store the accuracy of data.

2. *Kernos*

Models in Keras can come in two forms – Sequential and via the Functional API. For most deep learning networks that you build, the Sequential model is likely what you will use. It allows you to easily stack sequential layers (and even recurrent layers) of the network in order from input to output. The functional API allows you to build more complicated architectures, and it won't be covered in this tutorial. Next, we add a 2D convolutional layer to process the 2D MNIST input images. The first argument passed to the Conv2D() layer function is the number of output channels – in this case we have 32 output channels The next input is the kernel size, which in this case we have chosen to be a 5×5 moving window, followed by the strides in the *x* and *y* directions (1, 1). Next, the activation function is a rectified linear unit and finally we have to supply the model with the size of the input to the layer Declaring the input shape is only required of the first layer – Keras is good enough to work out the size of the tensors flowing through the model from there. Next we add a 2D max pooling layer. The definition of the layer is dead easy. We simply specify the size of the pooling in the *x* and *y* directions – (2, 2) in this case, and the strides. That's it. Next we add another convolutional + max pooling layer, with 64 output channels. The default strides argument in the Conv2D() function is (1, 1) in Keras, so we can leave it out. The default strides argument in Keras is to make it equal ot the pool size, so again, we can leave it out. The input tensor for this layer is (batch size, 28, 28, 32) – the 28 x 28 is the size of the image, and the 32 is the number of output channels from the previous layer. However, notice we don't have to explicitly detail what the shape of the input is – Keras will work it out for us.

IV. RESULTS

Parameters	**Excepted Result**	**Actual Result**
Satellite Images	Already present on Google earth	Take the satellite images from Google earth
Accuracy of model	High	High
Loss of data	Less	Less
Execution Time	Less	Less

Table No. 4.1 Test Cases

While designing application, there are few major tasks which are considered so that there should not be any loophole in the proposed system. First task which is to solved is to collect datasets of satellite images and we are successfully in finding dataset of satellite images. Another task is to check accuracy of the trained model and in our application, we are successful to achieve high accuracy of the model. Next task is to deal with loss of data when model is trained and our application do not loss so much data. The Execution time is depend only on the training of model, so if model is easy trained in less time so execution time of application will be less.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 248, 248, 64)      640
_________________________________________________________________
activation (Activation)      (None, 248, 248, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 124, 124, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 122, 122, 64)      36928
_________________________________________________________________
activation_1 (Activation)    (None, 122, 122, 64)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 61, 61, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 59, 59, 64)        36928
_________________________________________________________________
activation_2 (Activation)    (None, 59, 59, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 29, 29, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 53824)             0
_________________________________________________________________
dense (Dense)                (None, 1)                 53825
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 128,321
Trainable params: 128,321
Non-trainable params: 0
```

Figure No. 4.1 CNN result

Above figure indicate the final result of the CNN method applied on the input satellite image. As it is mentioned in the above section that we are using three rounds of CNN methods and the reason behind using only three rounds is that to decrease execution time of application because training of model takes so much time. So, in above figure, there are three columns which are layer, Output Shape and parameter found. In layer, there are conv2D layer, Activation layer and MaxPooling2D layer are present in each round. So, in the very first round of CNN result has found 648 parameters from the original image and in the next layer, conv2D layer has found 36928 parameter from the output of previous round and at the last round of CNN method, again 36928 parameters are found and as a result of this rounds, dense layer gives 53825 parameters as a final result of CNN. As a final result, we are getting total number of parameters as 128,321 which all are trained parameters. So, through these parameters, we are trying to find our major factors like agricultural field, source of water bodies, source of electricity and roof tops.

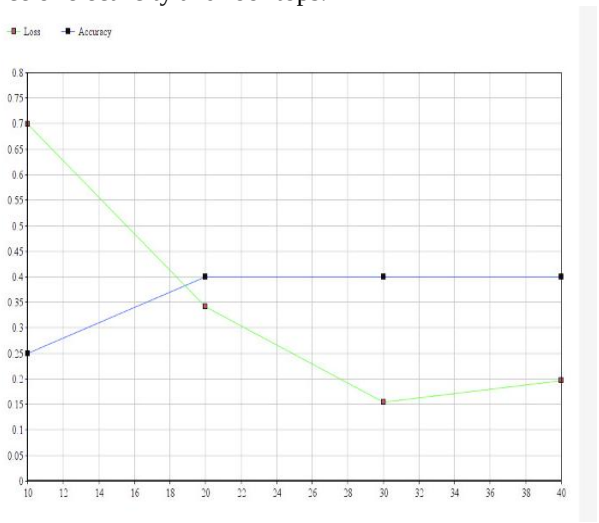

Figure No. 4.2 Graph of Accuracy and loss

This graph defines relationship between accuracy and loss of the model. This graph plot by using the points generated at the time of training of model. The graph indicate that line of loss decreases and line of accuracy remains constant. So, from this graph it can be say that the model trained is accurate with very less loss of data.

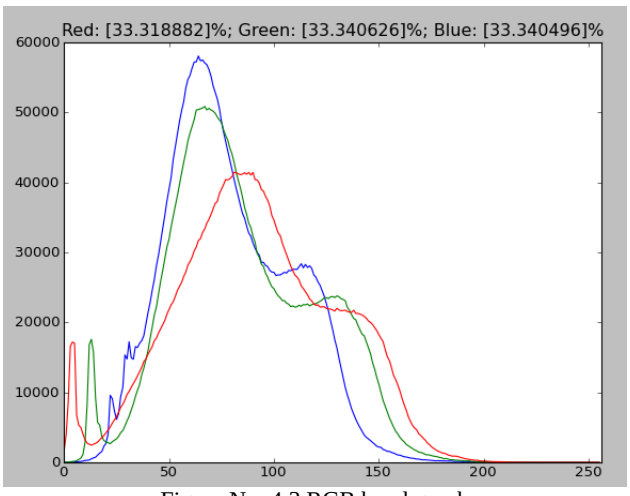

Figure No. 4.3 RGB band graph

This intensity graph is generated to check accuracy of the model prepared by the CNN method. This graph indicate color intensities present in the input satellite image. Through this graph, it is easier to say how many threshold factors are found in the input image.

V. CONCLUSION

To predict status of a satellite image, we have use preprocessing of a input image and in the training and testing, the dataset of satellite images were used to train the model. For training and testing of models CNN method is used where only 3 rounds are sufficient to predict accurate result of socio-economic status. In this way, we have successfully predict socio-economic status of a input image.

REFERENCES

[1] Pandey, S. M., Agarwal, T., & Krishnan, N. C. (2014). *Multi-Task Deep Learning for Predicting Poverty from Satellite Images.* Ropar: The Thirtieth AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18).

[2] OSHRI, B., HU, A., ADELSON, P., & LOBELL, D. (2018). *Infrastructure Qulity Assessment in Africa using Satellite Imagery and Deep Learning.* Stanford: Association for Computing Machinery.

[3] Perez, A., Yeh, C., Azzari, G., Burke, M., Lobell, D., & Ermon, S. (2017). *Poverty Prediction with Public Landsat 7 Satellite Imagery and Machine Learning.* California: 31st Conference on Neural Information Processing Systems.

[4] Babenko, B., Hersh, J., Newhouse, D., Ramakrishnan, A., & Swartz, T. (2017). *Poverty Mapping Using Convolutional Neural Networks Trained on High and Medium Resolution Satellite Images, With an Application in Mexico.* California: 31st Conference on Neural Information Processing Systems.

[5] https://youtu.be/QfNvhPx5Px8

