REPORT

ON

Land Use Land Cover classification and change prediction using Artificial Neural Network in python.

Submitted in the partial fulfilment of the requirements set by National Remote Sensing Centre, Hyderabad- ISRO and University of Mumbai

By

Dubey Amit Radheshyam

Under the guidance of

Mr. Hariesh Pesara

(Scientist/ Engineer 'SE'

Training Education Group

Training, Education & Outreach Group

Management Systems Area (MSA)

NRSC, HYDERABAD)

Fr. Conceicao Rodrigues College of Engineering

Bandra (west), Mumbai - 400050

University of Mumbai

Year 2020-21

Approval Sheet

This is to certify that the report entitled “Land Use Land Cover classification and change prediction using Artificial Neural Network in python” is a bona fide work of “Dubey Amit Radheshyam” submitted to National Remote Sensing Centre, Hyderabad- ISRO and University of Mumbai in partial fulfilment of the requirement for the completion of student project Internship.

Signatures of respected guide and others to be added here.

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Dubey Amit Radheshyam)

Date:

Content

Abstract

The report summarizes the research and solution of a problem statement given to me during my internship whose objective was to make use of Artificial Neural network and other Machine learning and deep learning methodology to perform Land use land cover (LULC) classification of a multispectral satellite image into various labels and predict the LULC changes. Hence the approach to the problem was divided into 2 stages, first being learning the concept of satellite imagery and techniques used to make the LULC maps and the second was to generate a methodology that could make use of the LULC maps as reference and predict the changes that could be seen in the coming future with an acceptable accuracy specifically using Artificial Neural Network. After thorough research and understanding the basics of remote sensing and the properties of a satellite image along with the different types of data that could be accessible and learning various methods in deep learning and machine learning that could be used to achieve the desire application, the objectives were achieved.

Stage 1.

1.1 Introduction:

This stage describes the process of understanding the satellite image data and executing the task of obtaining a classified LULC map. It involves the method of retrieving satellite image from accessible sources, pre-processing the obtained multiband satellite image and finally classification of the same by the means of machine learning techniques to certain classes and labelling them. Description of what all methods were tried and how they were not feasible is also mentioned.

1.2 Objectives:

1. To understand the basics of satellite imagery.
2. To learn the methods that can be used in python to retrieve and pre-process the data for further classification.
3. To learn the methodologies used for classification of satellite images.
4. To obtain a LULC map using ANN in python.

1.3 Understanding and accessing the data:

First most thing is to understand what are we dealing with hence understanding the data becomes a crucial part.

At the initial days of learning and trying methods to access and process satellite images various data from LANDSAT 7, LANDSAT 8, MODIS MCD43A4, SENTINEL1 and SENTINEL 2 were explored.

But for the final results the data from 10 m resolution Sentinel-2 L1C images were used.

Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

1.4 Descriptions of various methods:

1. Convolutional Neural network:
   After an initial research I got to know that the CNN is a very handy tool for unsupervised classification of images, so I tried to find and understand how I can apply it for LULC classification. The major drawbacks were that It required huge amount of training images for better results (in the range of 1000s) which was an issue with respect to accessibility and to compute these many images was taking way long than expected and hence required a better system. Yet after reducing the number of training data to 25 images of each class and using data augmentation I achieved an accuracy of just 58.7% and majority of classes were unclassified. Hence, I had to look for alternate methods.
2. Semantic Segmentation: Pixel based semantic segmentation with the combination of CNN for automating the setup was used which was based on UNET model, this solved the problem related to requirement of more data for better results as it could solve and give good results with a better accuracy. But the complexity of CNN with Semantic segmentation made it very heavy and time consuming to perform. When I tried with a smaller area satellite image(multispectral) the accuracy was also not up to the mark as shown below.

3. Orfeo toolbox: This made the process very easy and understandable, but for using this we needed to download the software and execute there itself which was making it too stiff and not very much in accordance to the requirement of the problem statement of the internship. Hence, I tried finding some alternate methods to satisfy the requirements.

1.5 Methodology adopted:

After understanding the problem statement and exploring the above-mentioned methods, I considered few points that were the basic requirements considering my computer system and availability of data to me. The redefined requirements were as follows:

- The method should take less time to compute.
- It should be less complex to be able to compute on a standard system.
- The handling of data should be easy and preferably on cloud.
- The method should provide good accuracy with less data requirement.

To satisfy all the above-mentioned requirement, I carried out research and tried a combination of few techniques to fulfil them. The final methodology adopted is as mentioned below:

1. Accessing the satellite data from sentinel hub:
   The sentinel hub is a great platform for obtaining satellite data of any region by providing the shape file (boundary) of the region. And by the addition of “sentinelhub” python package the task gets very easy and works on by just creating an id in the

same and adding credential in the python code. We can create a research purpose free id at sentinel hub through ESA OSEO.

2. Preparing the data:
   a. Input the geographical shapefile (.geojson file) of the required area of interest.
   b. Split it into smaller non overlapping rectangular tiles/patches for ease.
   c. Filling of the Patches with data namely L1C custom list of bands [B02, B03, B04, B08, B11, B12] which corresponds to [B, G, R, NIR, SWIR1, SWIR2] wavelengths, cloud probability map and cloud masks from sentinel hub services.
   d. Adding some additional information like NDVI, NDWI, Euclidean NORM calculated by combination of bands.
   e. Also adding the reference map raster to the patches for supervise learning procedure.
   f. Finally converting the patches to raster and saving the file.
3. Pre-processing the data:
   a. Removing the too cloudy scenes by checking the ratio of valid data for each patch keeping only the frame with <20% cloud cover.
   b. Concatenate the data (bands and additional information added) into a single feature.
   c. Perform temporal interpolation as due to non-constant acquisition dates of the satellites and irregular weather conditions missing of data is possible hence filling the gap by linear interpolation method.
   d. Execute the erosion to remove the artefacts with a width of 1 pixel and the edges between polygons of different classes.
4. Preparing for training:
   a. Random spatial sampling by a subset of pixels from a patch.
   b. Splitting into training and testing set (80% and 20% respectively).
5. Model construction:
   The training data is converted into a 5-dimensional array with parameters of number of patches(a), the number of resampled time frames for each patch(b), width(w), height(h), and depth(d) of each patch. But the classifier only takes a 2D array so this has to be converted into the same which can be done by simply reordering the pixels and the feature array becomes 2D array with 2 inputs as (a*w*h, b*d). This allows to make a prediction on new data of the same shape.
   The classifier options available were many to name a few SVM, LightGBM, ANN, etc. but out of all of it the LGBM was the quickest as tested by experimenting three of them one by one.
   Now since the data is shaped in the required 2D array format no such changes are required for implementing any of the classifier. The standard format is used, but the for improving the results we need to use the hyper parameter tuning which takes time but automates the job of tried and error for obtaining best accuracy, in my work in used grid search for the same.
6. Validation:
   The training data is further divided into train and validation set (80% and 20%), the validation is done by calculating confusion matrices, receiver operating characteristic curve and feature importance map.
7. Visualization and saving the labelled image.
   The image is plotted using matplotlib library and the classified map is saved and exported using gdal library. All the validation graphs are also plotted using matplotlib and finally to visualize the actual difference between classified image and ground truth we use matplotlib.

Data used for training the algorithm was of Slovenia region, and after the training was done with a very an accuracy of 93.3%, I used this data for training because of its free availability of reference classified image and which was done using ground techniques so the classification validation and testing was more accurate.

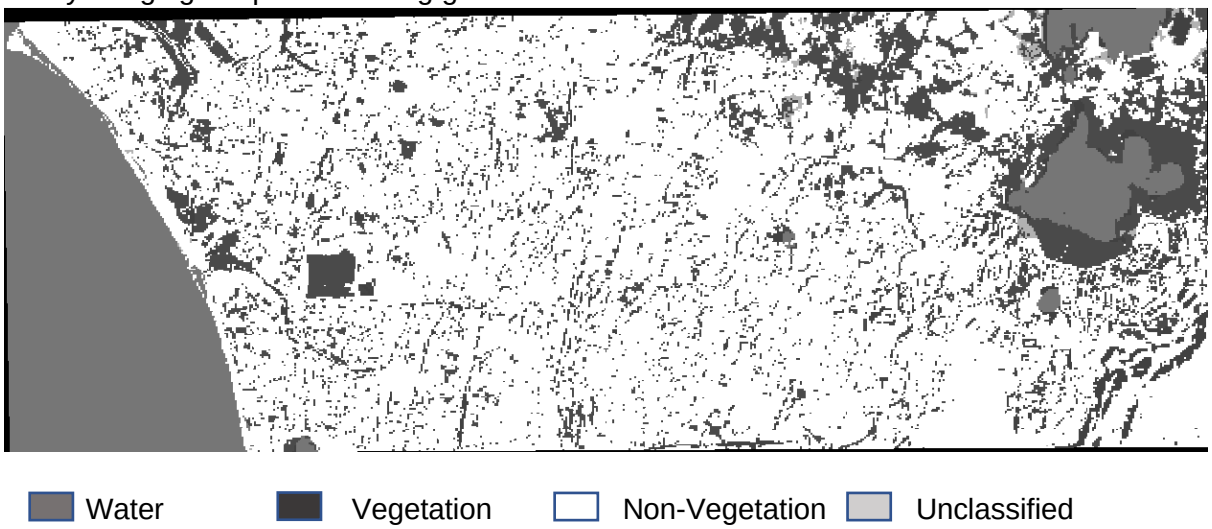
1.6 Implementation and Results:

1. After the training was done, I used data of MUMBAI SUBURBAN Region which is my home town for prediction(classification) as per the requirement of the project. The region of interest of Mumbai Suburban region of 58.4Km^2, of Sentinel-2 L1C was downloaded of the year 2017.

2. The True colour satellite image on 1/07/2017 of the same was retrieved as shown below.

3. The plot of temporal mean of NDVI and NDWI is obtained as below:

4. After this the prediction starts and for the best of accuracy as mentioned above, it took **27mins 20 seconds** to classify only.
5. The classified image (plotted) is obtained as shown below and was exported as a .tiff file by merging the patches using gdal.



1.7 Future scope:

- Once the model is trained which in my case was trained with 5 years of database it can be used to for classification of any other region now, without the reference map and just by getting the shape file of the region of interest.
- The procedure is extremely simple and user friendly. The user just needs to add shape file of the required region to the python notebook where it is prompted and execute the code. It will ask to choose the patch of interest and the number of surrounding patches. Once that's done, in about 30 minutes the results will be displayed on the screen as a plot and the predicted classified file will be saved to the mounted drive location. (Considering the user is using it on Colab and with GPU server.)

1.8 Resources utilized:

1. List of python packages used are:
   - Geospatial data handling: gdal, eolearn, sentinelhub.
   - Machine learning: sklearn, keras, lgb.
   - Visualization and data handling: numpy, matplotlib, geopandas.
2. System used: Cloud GPU server through google Colaboratory.

Stage 2

2.1 Introduction:

This stage describes the process of building an Artificial neural network system combined with cellular automata to takes LULC map of 2 different years of the same region and a few explanatory maps to predict the future trends or changes in the region and visualize those changes by generating a map of the same.

2.2 Objective:

The major objectives of the problem statement in the internship were as follows,

1. To obtain the LULC maps of 2 different years of the same region and explanatory maps for the same.
2. To obtain a transition potential matrix for an input to the cellular automata by making use of ANN.
3. To generate the predicted LULC map that depicts the possible change that could happen in the coming years.
4. To validate the results obtained after the simulation to check the accuracy of it.

2.3 Methodology:

The prediction model is a combination of ANN and Cellular Automata for simulating the changes and predicted a future LULC change map. As it is known that a classic cellular automata works on rule-based model which has to be predefined by the user for it to function, but here in this method instead of using explicitly defined rules it makes use of a model which is developed by the Artificial neural network that gives the transition potentials(probabilities) of each pixel using information that are input. This way the process becomes automated as the rules are now implicit as received by the ANN fit model.

2.4 Workflow:

1. Input LULC maps of 2 different time periods of the same region, with the same shape, band count and classes.
2. Input the explanatory maps like the distance from roads, distance from main city, distance from water bodies etc. as per the availability of them. These maps are basically made up intensity maps with values varying from 0-1 for ease where 1 indicated the most intensity that means the closest (or within) the parameter (roads, river, city). This helps in understanding the region well and gives the ANN model better chance of prediction the transition potentials of the pixels based on these factors. And mostly it is observed that the more the information the better the result.
3. Converting all the files into raster and by area calculation of each classes develop a transition matrix from one class to other.
4. Building an ANN model to take in the raster of input data, the transition map.
5. Training the model.
6. Predicting for getting the transition potential matrix.
7. Running the simulation using cellular automata which works on the rules as per the transition potential received from the prediction model.
8. Finally saving the Predicted LULC map and checking the accuracy using the Kappa coefficient and Accuracy (if a map of the predicted period is available).

2.5 Implementation:

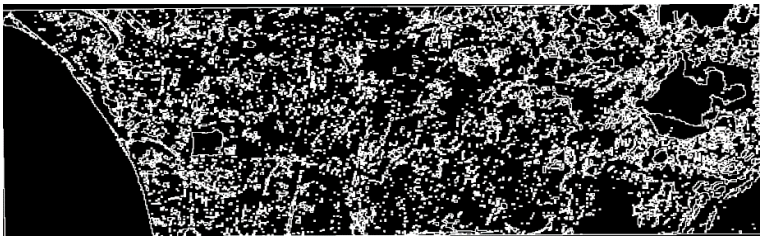
1. Data used:
   For this purpose, I used LULC maps and explanatory data of Mumbai suburban region that were classified using the classification algorithm as mentioned there. As the explanatory maps were readily available and It made me test the algorithm in a faster way. The LULC maps and explanatory maps used are shown below.

2. Preparing the data and normalization:

Proximity from City Centre

Proximity from the sea

Slope



Creating and storing the raster, converting to the required inputs to the model like stacking the arrays of raster data and also checking the size and specification of the data for avoiding discrepancy.

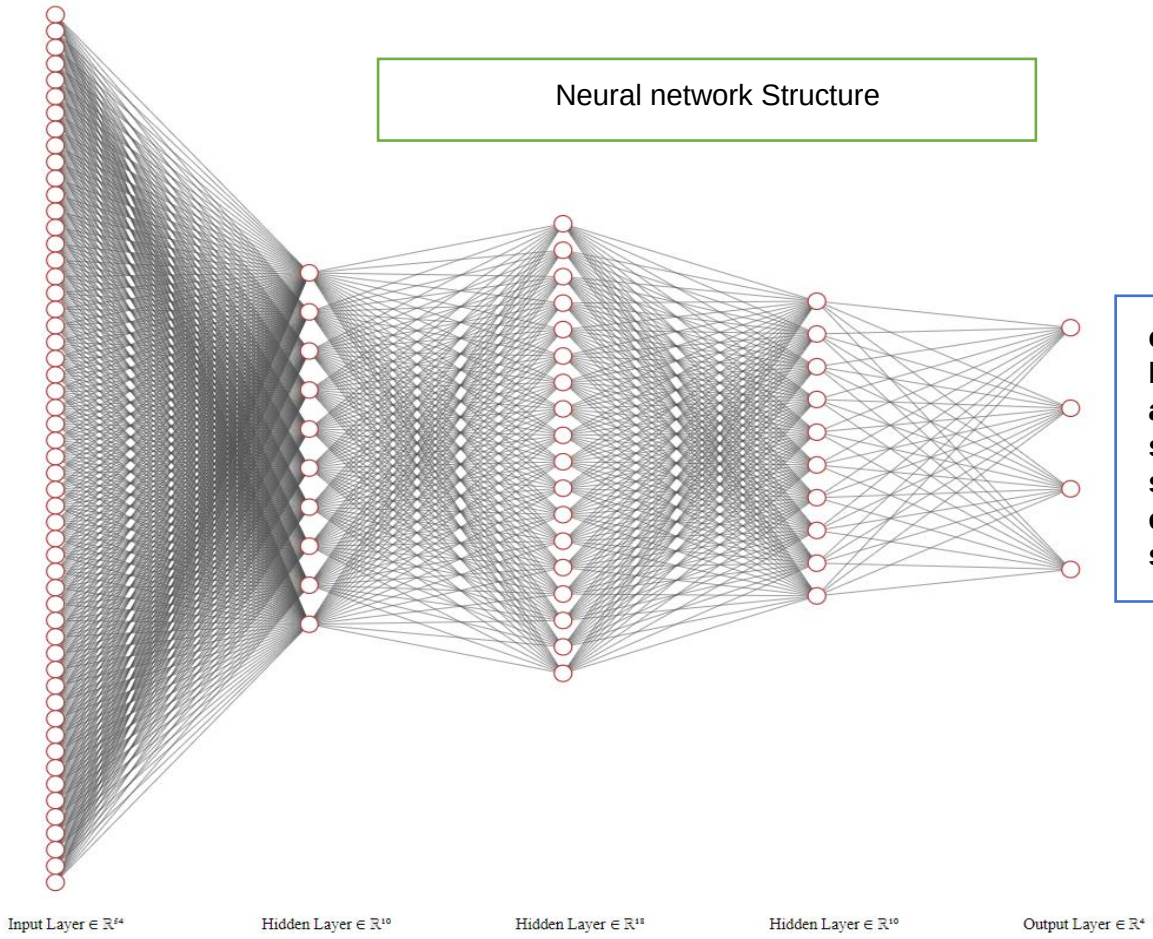
Finally Normalizing the factor data for efficient training and faster result, a simple linear normalization was used by subtracting the mean of data values from the data value of the specific pixel and dividing the difference by standard deviation of the data values.

3. Transition Matrix: The transition from class to class from comparing the area growth (change in pixel classes) of the two LULC maps are shown below.

```
Transition Matrix
[[8.02538630e-01 1.56444390e-02 1.81800119e-01 1.68121170e-05]
 [5.96749128e-02 9.33973792e-01 6.34633831e-03 4.95710862e-06]
 [2.86022891e-01 3.57781026e-03 7.09174170e-01 1.22512897e-03]
 [5.33333333e-02 7.61904762e-03 7.20000000e-01 2.19047619e-01]]
```

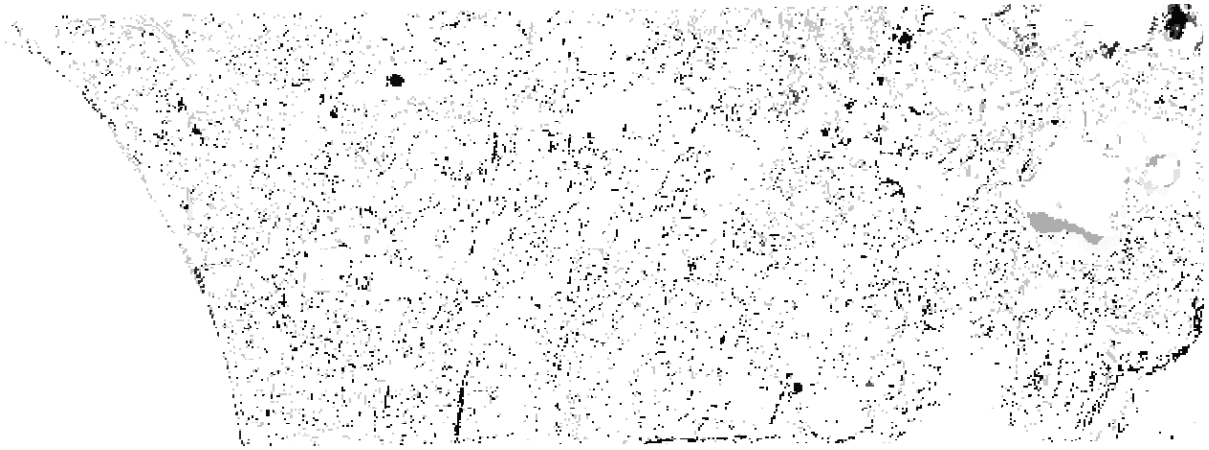
4. ANN model:
   The input data to the model was the collection of the data as mentioned above and the output from the model will be the transition map and transition probability matrix for each pixel.
   The structure of it consist of an input layer, an output layer and defined number of hidden layer (discussed later).
   The number of input layers to the neural network depends on the number of LULC classes(A), total number of factors maps (considering all are single band maps) (B)and the neighbourhood size(C) which can be varied as per the required accuracy.
   Therefore, the number of neurons in the input layer is given by,
   Input layer neuron count= $\mathbf{(2xC+1)^2x(A+B-1)}$;
   In this case it was $(2*1+1)^2(4+3-1)$ = **54**
   The number of output neurons are same as the number of classes, which in this case was **4**.
   For the activation function, I have used Sigmoid function which is a non-linear activation function.
   It uses the backpropagation with momentum for learning and updating the weights.
   The learning rate was set to **0.1** and the momentum at **0.1** for this case.

Neural network Structure



For training the data set was divided into 2 parts training (80%) and validation (20%) and the data were chosen randomly.

Transition Map



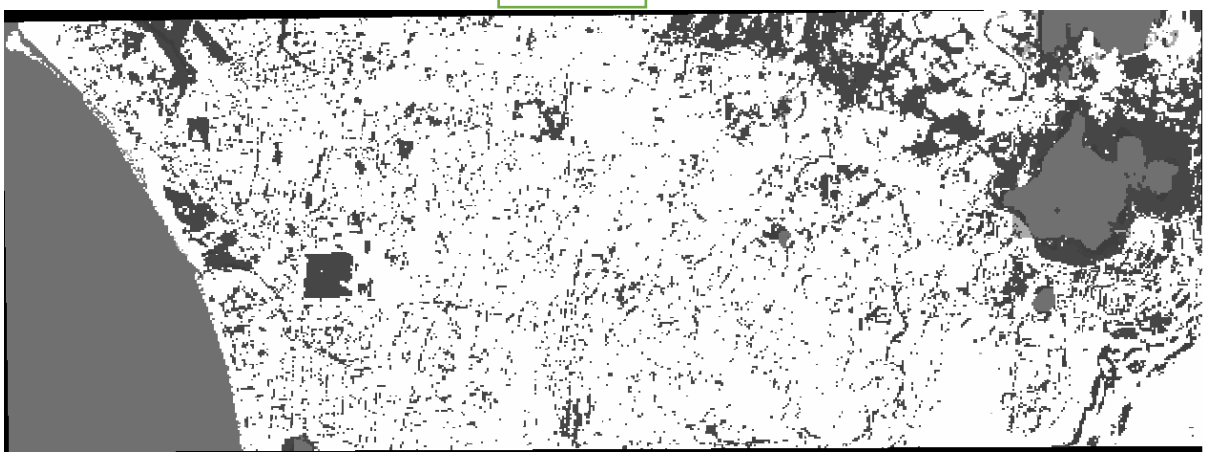
5. Cellular Automata Simulation:

The model takes the inputs as the Initial period map and explanatory maps in raster form along with the transition probabilities got from the ANN model for using as a rule function for each pixel. So, it doesn't use explicit rules but the rules generated by the ANN model. The simulation works as follow:

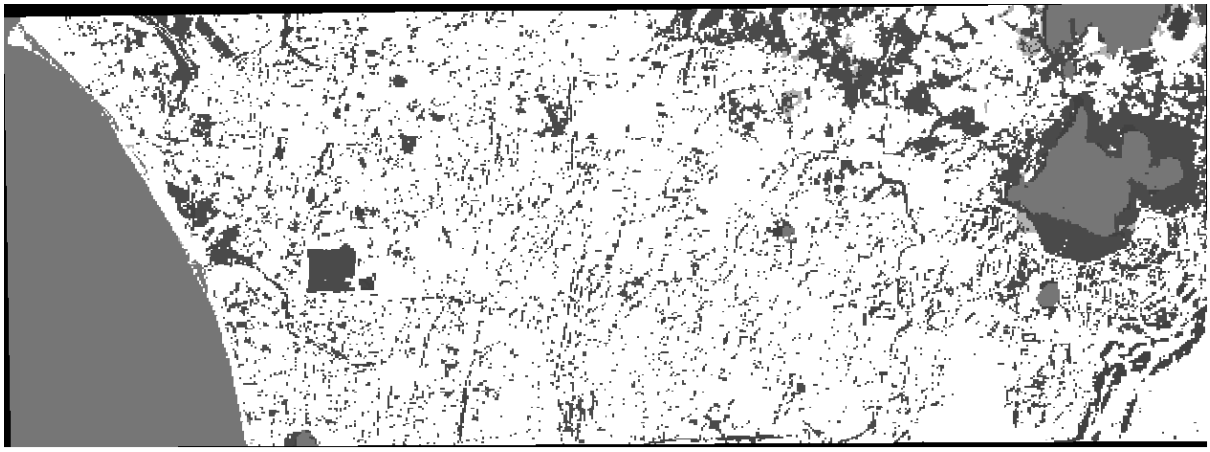
- It takes transition probabilities and calculates count of pixels that has to be changed for every transition class.
- It scans pixels of the raster and its neighbours and obtains transition potentials of every transition class.
- It makes a raster of the most probable transitions. The pixels of the raster are the transition class with the biggest potential of transition it is an auxiliary raster that is used in the next stage.
- For each transition class the model search in the raster of the most probable transitions. Pixels with the greatest confidence changes the category of the pixels.
- Finally, the output is exported, which is the predicted LULC map.

The final output map is basically the map of the difference between the periods of the input map. So, in this case it predicted 2021 map after taking in year 2017 map as shown below. And so, to predict for the next year which will be 2025, the input will be the predicted 2021 map. Which is made easier in my module by just changing the number of iterations of the simulation.

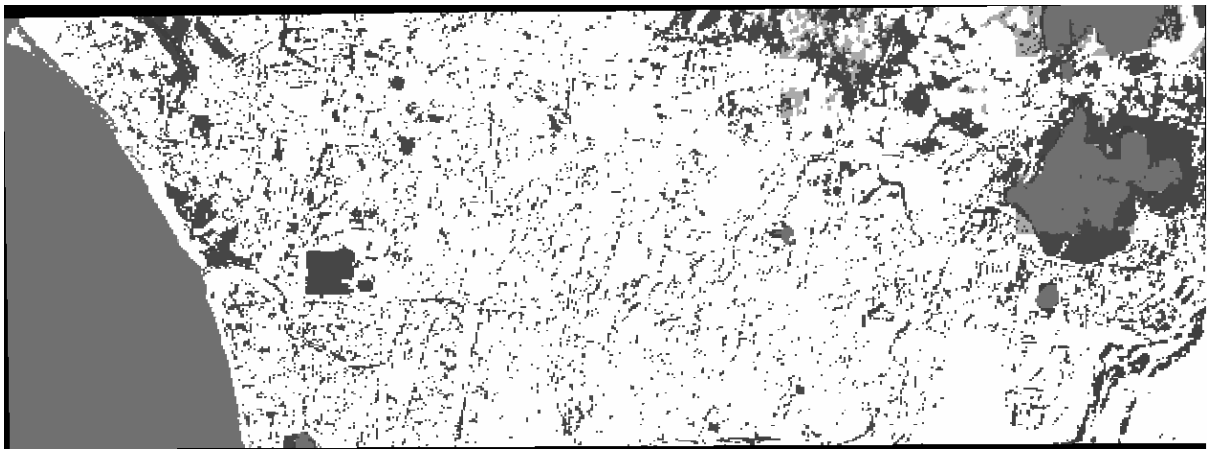
2017



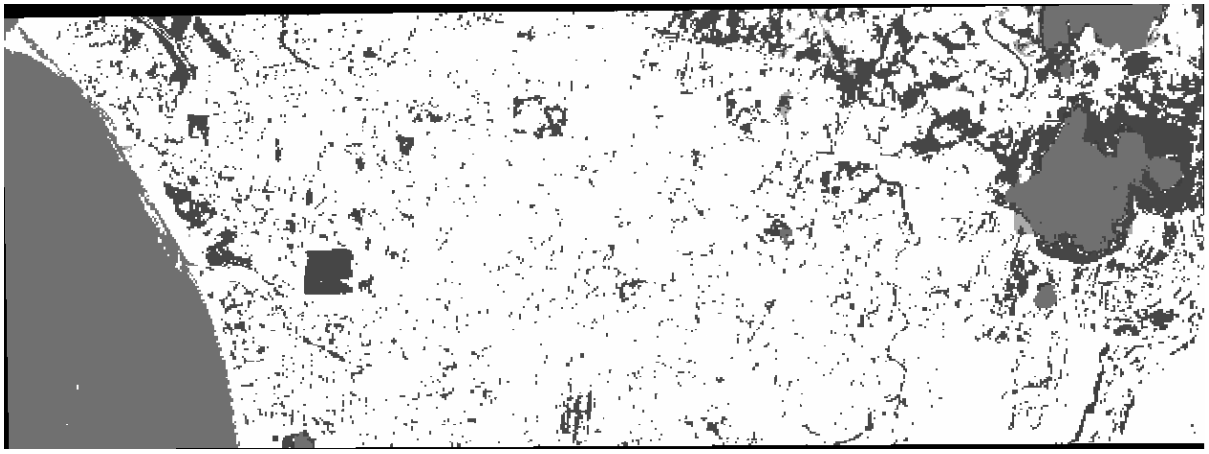
2021



Predicted 2021 Map



Predicted Future Map 2025



2.6 Results:

By keeping the epoch for the ANN model to 10 and the size of neighbouring cell to be 1, I got the accuracy of **74.3**% (comparing 2021 predicted map to the existing 2021 map) and the kappa coefficient of **76.58**% which was the best result I got after tuning the parameters.

Total runtime took **43mins** to compute on a 16GB RAM and 4GB GPU server.

As seen from the predicted map of 2025 it is pretty much understandable that the built-up class is growing by a huge percentage and that takes land from agricultural and mostly barren and forest land covers.

2.7 About the Script and usability:

- The hidden layers of the ANN model is still not fixed but after performing few iterations by tried and error method some results were obtained and the one with maximum accuracy has to be selected, to skip this part I made use of hyper

parameter tuning which automated this process and took a lot of time. But finally, the satisfactory accuracy as mentioned below is obtained for a 3 layer [10,18,10] neurons each hidden layer.

- For the ease of understanding the python module is divided into a few function bases scripts which can be called from a python notebook at one using file handling very conveniently and a layman user doesn't have to go through the whole thing but just run the notebook on the system and after the computation is completed the files will be automatically saved into the current directory.
- For my case I computed everything on the cloud using the servers provided by the google Collaboratory platform. So, the problem of need of a high-end system is vanished as the computation can be done on the browser. If at all the user needs to use the local system and considering the system to be slow, the module will still run because of lack of use of heavy packages.
- Even though it is not yet a GUI based module but still the python notebook is so interactive and easy in terms of workflow that everything becomes simple and understandable. The processes are prompted in the screen with timeline as well.

2.8 Resources utilized:

3. List of python packages used are:
   - Geospatial data handling: gdal.
   - ANN and CA construction: NumPy.
   - System used: Cloud GPU server through google Collaboratory

2.9 Future Scope:

- Due to lack of resources at my end I couldn't create a GUI for the same, but to cover that my runtime is very interactive. Perhaps a GUI will be a great addition to the module and that is something that I will look forward to develop soon after I have access to a better system.
- The explanatory maps play a major role in the prediction as it is again a supervised learning method hence the accuracy of maps should be increased. Since I used data from an open source platform, it will be a better act if I get to make or use authentic factor and input LULC maps.
- The accuracy can be increased further by increasing the epoch which will increase the runtime as well.

Conclusion

After the spending more than 3 months on this, I have finally achieved the desire results in accordance to the problem statement. The learning was challenging as Remote sensing and GIS was absolutely new to me and the application of deep learning and artificial intelligence was not easy to understand. But now I have understood the importance of Remote sensing especially LULC maps and predicting the changes of the same, I have read about many applications of this for the benefit of our country and even beyond the national boundaries. And the use of Artificial Intelligence makes the task faster and accurate, which could be a great help in dangerous situations like the recent Uttarakhand glacial burst calamity. These kinds of incidents can be prevented if an accurate and faster model is developed to forecast the probabilities of them to happen.

My work is based on simple python scripts and no such heavy packages that could burden the user to download them. It is user friendly as it simply needs two python notebooks to run one after the other to finally get the output of before mentioned accuracies in about 45minutes.