SEVAS Publishing

Computing Technology Research Journal
Compute. Tech. Res. J. (2022) 1(1):22-29

DETECTION OF CLOUD SHADOWS USING DEEP CNN UTILISING SPATIAL AND SPECTRAL FEATURES OF LANDSAT IMAGERY

M.S. Antony Vigil, Aashna Chib, Ayushi Vashisth, Tanisha Pattnaik
Department of Computer Science and Engineering. SRM Institute of Science and Technology, Chennai, India
*Email: antonyvigil@gmail.com

Received: 10th September 2022, Accepted: 3rd October 2022 and Published: 3rd October 2022

ABSTRACT

Aim: The proposed work emphasizes here on detection of cloud shadows using Deep CNN (Convolutional Neural Networks) utilizing spatial and spectral features of Landsat imagery.
Results: In the current study deep CNN Algorithm is used for cloud and its shadow detection. We used python libraries to create a CNN. Fourier transformation is applied on that array to transform as per their requirements.
Conclusion: Using the Deep CNN algorithm, we were able to combine the whole input image to get multilevel features. Deep CNN does better image processing and semantic segmentation when compared with existing fuzzy-c and f-masking.

Keywords: Convolutional Neural Network (CNN), Cloud Detection, Semantic Segmentation, Satellite Imager, Landsat, Fuzzy-C.

HIGHLIGHTS:

1. **An improved approach using Deep CNN (Convolutional Neural Network) does better image processing and semantic segmentation when compared with existing fuzzy-c and f-masking.**

INTRODUCTION

In the suggested system, Deep Convolutional Neural Network Algorithm for the detection of cloud and shadows is used. CNN helps in efficient image processing and sematic segmentation. TensorFlow library is used create a CNN with approximately 13 hidden layers. Landsat images are then passed as input and each pixel of the image is categorized into 3 categories. After training the model, new Landsat images are fed and lastly whether the results are appropriate or not are checked. The procedure for detection of cloud and the shadow involves five modules. The five modules for cloud and cloud shadow detection are as follows:

(1) Preprocessor module;
(2) Training module;
(3) CNN (Convolutional Neural Network) module;
(4) Upsampling module;
(5) Pooling module

Preprocessor Module

The preprocessor module is the very first step involved in the process of detection of clouds and the shadows. This module is used for the preprocessing purpose. The preprocessor module takes the Landsat images as input. After the input is taken the module processes the raster images and sets the radio frequency range (also known as Band VI). The images are then resized and standardized as per the requirements. After the images are resized, they are then flipped and rotated to make sure that all images are oriented properly with same size and split bands. The flowchart for the module is shown in Fig. 2.

Training Module

The meaning of training a system is to determine upright values for every weight and thus upright values for every bias from instances that are labeled. Here we attempt to build a system by studying and learning through various instances and then tries to seek out a system that reduces the loss or make it minimum. Degradation or Loss indicates a poor projection or prediction i.e degradation or loss may be a number or a value that indicates how bad the system forecast was on one instance. Suppose predictions given by the model are perfect, then there's no loss but if the predictions aren't perfect, the loss is bigger. The aim of model training is to look for a pair of biases and weights that provides lower loss, in overall, for each example. Figure. 3. showcase a system on left side that has higher loss and a model on the right side that has lower loss. Remember the subsequent about the figure:

The aim of training module is to create an accurate model that answers our questions correctly most of the time. Training module trains the system to get positive results. Landsat images are sent as input and then are processed using CNN (Convolutional Neural Network). CNN further gives its feedback and then more Landsat images are trained.

Loss Function

Mean square error (MSE) is the most common function used for measuring loss. It is the average of all squared losses per instance over the entire dataset. The Mean Square Error is calculated as below:

Mean Square Error $=1N\sum(y,z)\in D(y-\text{predicted}(y))^2$

Where y is the set of features, z is the label, predicted(y) is the function consisting the weights and biases along z, D is the set of data that contains various labeled examples and N stands for number of examples in D.

CNN Module

CNN model takes input in the form of Landsat images. 13 hidden layers are used where each layer extracts some information and cluster it. The activation function of CNN is leaky ReLu. ReLU stands for Rectified Linear Unit which is an operation that introduces non-linearity. The outcome for ReLu is $f(y)$ = maximum(0,y). Theta represents weight of each node. The output will be segmented into 3 categories: cloud, dark cloud shadow, and light cloud shadow. Image is considered as a 2d array. Values between 0 to 1 or 0 to 255 based on how it is processed are filled. To do so, a portion of the mirror image is added to boundaries, where some existing data is taken from the 2d array right to left and extra boundaries of 6x6 are filled. Fourier transformation is applied on that array to transform as per their requirements. Using the Deep CNN algorithm, we will be able to combine the whole input image to get multilevel features. Deep Convolutional Neural Networks are typically the modified networks of neural, that apply various findings and algorithms in order to adjust the weights and biases to get low-cost function.

Up-sampling Module

Up-sampling module is used to bring back the reduced size of output image to what the size of the image was before initially. The resultant image that we get after passing image from CNN layer is compressed. In order to get the initial input size image, Up-sampling is used. With Up-sampling, the 2D representation of an image is kept intact while only the spatial resolution is increased. It is used to remove a phenomenon called pixilation effect. It is caused when relatively large frame displays an image of less resolution. It is also used to zoom in on a region of an image which is small. Image is considered as a 2d array. Values between 0 to 1 or 0 to 255 based on how it is processed are filled. Imagine we have a 2d array of dimensions 3x3 and we want 6x6, so we need to remap the array while filling the gaps. To do so, a portion of mirror image is added to boundaries, where some existing data is taken from 2d array right to left and extra boundaries of 6x6 are filled. Fourier transformation is applied on that array to transform as per their requirements. For balancing they rotate 2d array, then apply operations on it. Finally, after the operations, we get the up- sampled image. The flowchart for the module is shown in Fig. 7.

Pooling Module

Input images are applied with filters that are learned by Convolutional Neural Network layers called convolution layers so as to make feature maps. The input is summarized by the presence of various features. Convolution layers are very effective and when they are stacked layers on the brink of the input to find out low-level features is also allowed. Also, it helps the layers that are deeper within the system to find out features that are more detailed, for example various shapes. The output of feature map records the precise position of features within the input which is the limitation of the feature map output as this means that tiny things or movements that are not important in feature in the input image will become a special or important feature map. Re-cropping, rotation, shifting, and other minor changes to the input image causes this problem. This problem can be solved by adding a pooling layer. Pooling Module reduces the feature maps proportions. Here the number of parameters that are to be found and the amount of estimation that will be done in a network is reduced by this pooling layer. The features present during a region of the feature map generated by the convolution layer is summarized by this layer. So now, Further operations are performed on these summarized features instead of the features produced by the convolutional layer. Adding of this layer makes the system more efficient and thus the system becomes more flexible to changes made the features position in the input image. The flowchart for Pooling module is shown in Fig. 7.

Also, compared to valid loss, train loss is rather steady. Emphasizing that the model is not taking large steps over the gradient curve to match with the dataset, which would have resulted in overfitting. More Epochs would have led to saturation of backpropagating weights and would have negligible effect on the weight of nodes present in network. Once the training ends, the model is used for predicting the masks on images from dataset. Below are few results which show the original image, ground truth and the predicted mask.

RESULTS

The Model is trained on train dataset for about 50 epochs with Adam optimizer at a learning rate of 0.01. This hyperparameter was decided based on few experimental epochs. Based on the graph in fig.9 which plots train loss against valid loss, it can be seen that over time, the model is much closer to ground truth and has avoided being overfitted or underfitted to the dataset used.

In Fig 10, the leftmost plot is the Greyscale version of image being provided as input to the model on which mask is to be predicted. The middle plot shows the ground truth for that input image based on training. The rightmost plot is the mask predicted by the model which needs to be as close as possible to the ground truth. Which as visually observed, stands true in this prediction. In another prediction shown in Fig 11, it can be seen that the prediction is way off from the ground truth.

Figure 11 depicts the cases where the model was not trained adequately for the given scenario due to lack of variety in dataset. The original input image for figure 11 has half of the image blocked and thus completely black, and for the other half where Landsat imagery of geo location is observer, the amount of clouds present is really low to the point where it can go unnoticed if not carefully observed. The Ground truth for the input image is also quite far from actual cloud presence as visually observed. Since such cases are low, the model didn’t have enough data and hence failed to properly predict the mask for the input image.

Figure 12 depicts a case where the original input image doesn’t have any clouds, but rather terrains which may be perceived as cloud due to the way they are observed. However as seen in the prediction, the model is able to discern such geographical terrain from clouds and so didn’t produce any masking which is the right prediction in this scenario. This till now has shown the model trying to catch up with ground truth but the next two predictions show visually that ground truth was not as accurate as the predicted mask. This further shows that the model was not overfitted to data and is able to predict masks for wide range of scenarios that can be seen for any given set of input images. Fig. 13(a) and (b) showcases the scenario mentioned above.

Perceived as cloud due to the way they are observed. However as seen in the prediction, the model is able to discern such geographical terrain from clouds and so didn’t produce any masking which is the right prediction in this scenario. This till now has shown the model trying to catch up with ground truth, but the next two predictions show visually that ground truth was not as accurate as the predicted mask. This further shows that the model was not overfitted to data and is able to predict masks for wide range of scenarios that can be seen for any given set of input images. Fig. 13(a) and (b) showcases the scenario mentioned above.

Since there’s lots of disturbance in climate and the occurrence of clouds over an area is sporadic, images were taken from different temporal points over the same spatial pocket on a fairly clear day and cloud heavy temporal points. Since it's hard to get the same level of surface reflectance throughout different temporal points over 12 hours, we need to assume that reflectance from the surface for both images should be about the same if there are about 16 days. This is under how a solo Landsat takes images over the same spatial pocket.

Before moving on to the conclusion, it is important to address the way the algorithm is being assessed with the reference dataset. Since the distortion is high and variability in cloud presence thickness was high across spatiotemporal planes, it would be hard, rather impossible to use much established traditional measurement like in situ measurement, as usually used in other remote sensing algorithms. Also, if we can't get a relatively cloud-free reference image temporally, then the outcome would be slightly affected by it.

CONCLUSION

By using deep CNN instead of fuzzy clustering, the image processing will be better with proper semantic segmentation. It will classify every pixel of the image for much more accurate detection and de nosing. It will take multiple bands as input for detection of cloud and its penumbra. Since the hidden information can be ruptured by deep CNN methods, it is expected that the proposed work should detect clouds and its penumbra in multi-band images. The proposed Deep CNN model proved to be mor efficient than existing models as shown in results and discussion, what was not mentioned there would be how efficient it was in terms of computational efficiency. For the Quantitative assessment of the proposed algorithm, histograms were generated and changes in histogram were noted as they dictate how efficient the algorithm was in the removal of clouds. However, there are few limitations of the proposed model. For one, it is hard to call this model, a universal model that can operate on all types of images taken universally. Though with enough variety in a dataset and retraining the model with higher quality and quantity dataset should be able to handle universal images. Gathering such a dataset is out of scope for what the objective was, and since Landsat images from Landsat 7,8 happen to be used widely, it is safe to form impressions around its efficiency based on its dataset. Overall, the proposed model has performed well under a variety of conditions and its ability to do so with less number of available channels while being able to account for spectral domains and spatiotemporal domains as well. To make this model more generalized, a lot more training will have to be done to increase in available ground truths for the model and improve its capability to detect and remove much thinner clouds as well.

REFERENCES:

1. https://doi.org/10.1504/ijdmb.2014.062887
2. https://doi.org/10.1186/2041-1480-2-3
3. https://doi.org/10.1186/1471-2105-11-588
4. https://doi.org/10.1093/nar/gki070
5. https://doi.org/10.1093/nar/gkv1344
6. https://doi.org/10.1371/journal.pone.0113859
7. https://doi.org/10.1109/TCBB.2017.2695542
8. https://doi.org/10.1038/s41598-018-33219-y
9. https://doi.org/10.1186/s12859-016-1160-0

FIGURES:

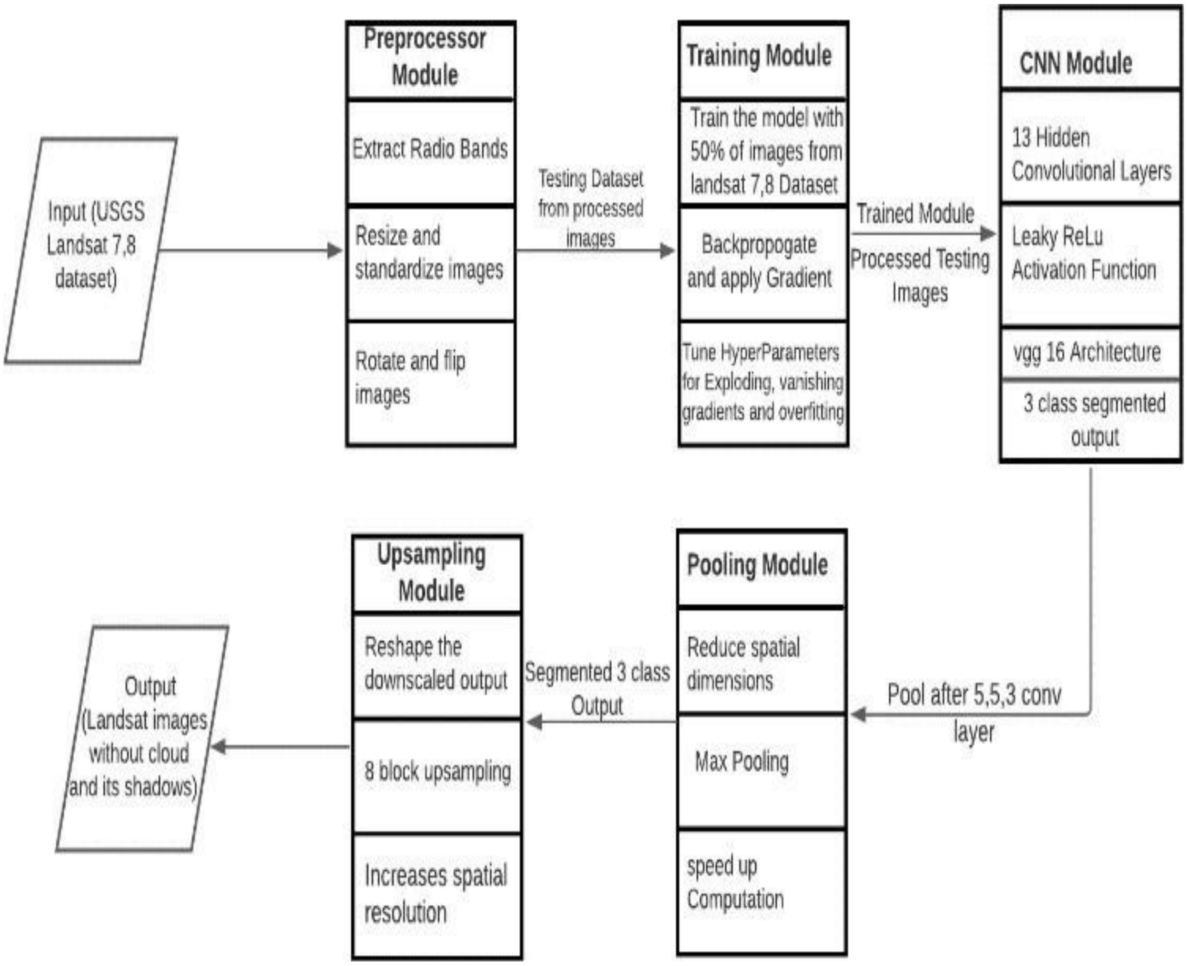
Input (USGS Landsat 7,8 dataset)
Preprocessor Module
Extract Radio Bands
Resize and standardize images
Rotate and flip images
Testing Dataset from processed images
Training Module
Train the model with 50% of images from landsat 7,8 Dataset
Backpropogate and apply Gradient
Tune HyperParameters for Exploding, vanishing gradients and overfitting
Trained Module
Processed Testing Images
CNN Module
13 Hidden Convolutional Layers
Leaky ReLu Activation Function
vgg 16 Architecture
3 class segmented output
Pool after 5,5,3 conv layer
Pooling Module
Reduce spatial dimensions
Max Pooling
speed up Computation
Segmented 3 class Output
Upsampling Module
Reshape the downscaled output
8 block upsampling
Increases spatial resolution
Output (Landsat images without cloud and its shadows)

Fig. 2. Flowchart of cloud and cloud shadow detection using CNN

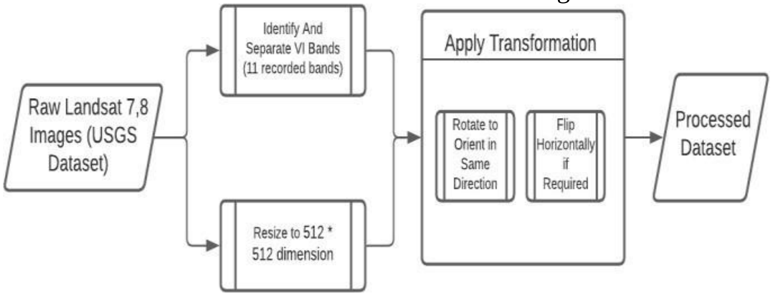
Raw Landsat 7,8 Images (USGS Dataset)
Identify And Separate VI Bands (11 recorded bands)
Resize to 512 * 512 dimension
Apply Transformation
Rotate to Orient in Same Direction
Flip Horizontally if Required
Processed Dataset

Fig.3. Flowchart for Preprocessor module

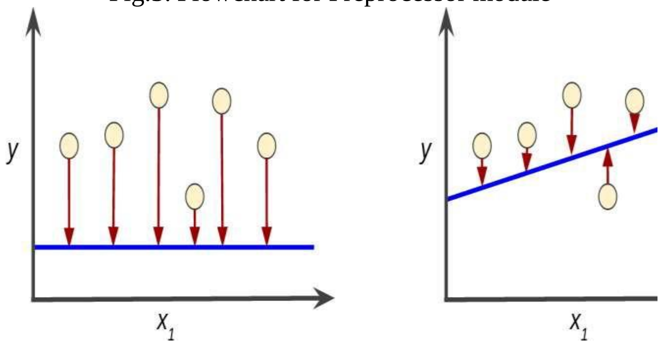

Figure.4. Model with higher loss in left side; Model with lower loss in right side

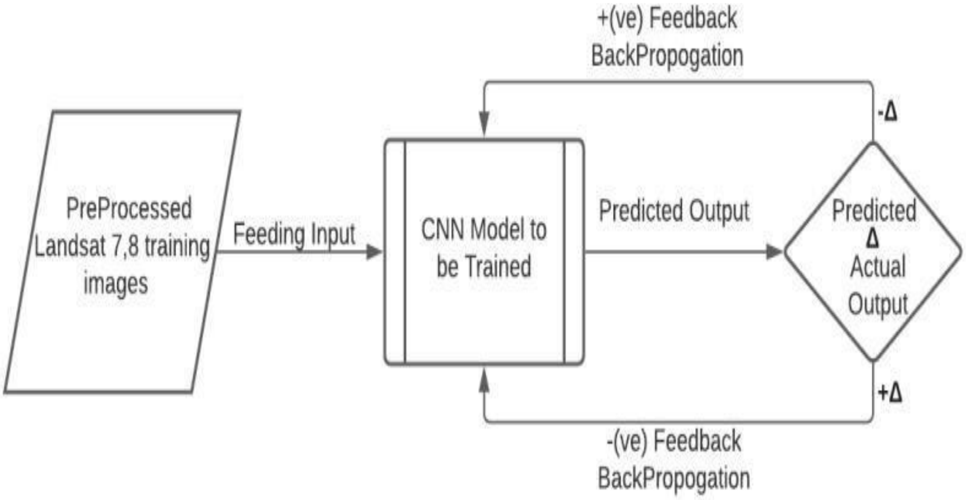

Fig.5. Flowchart for Training Module

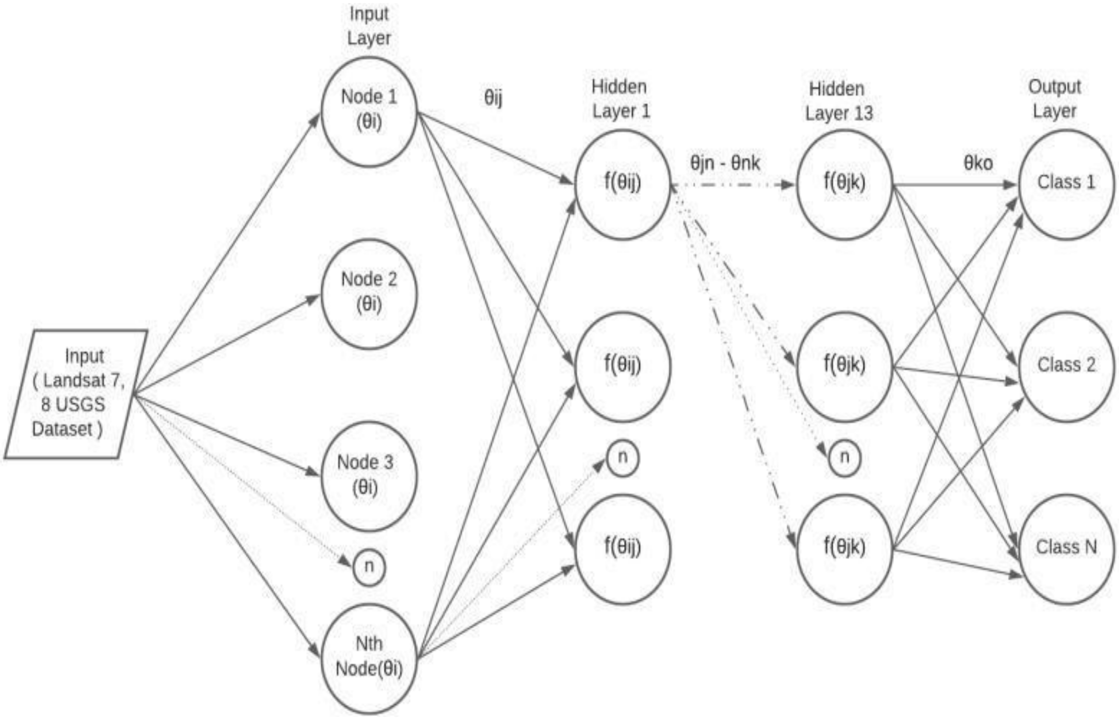

Fig.6. Flowchart for CNN Module

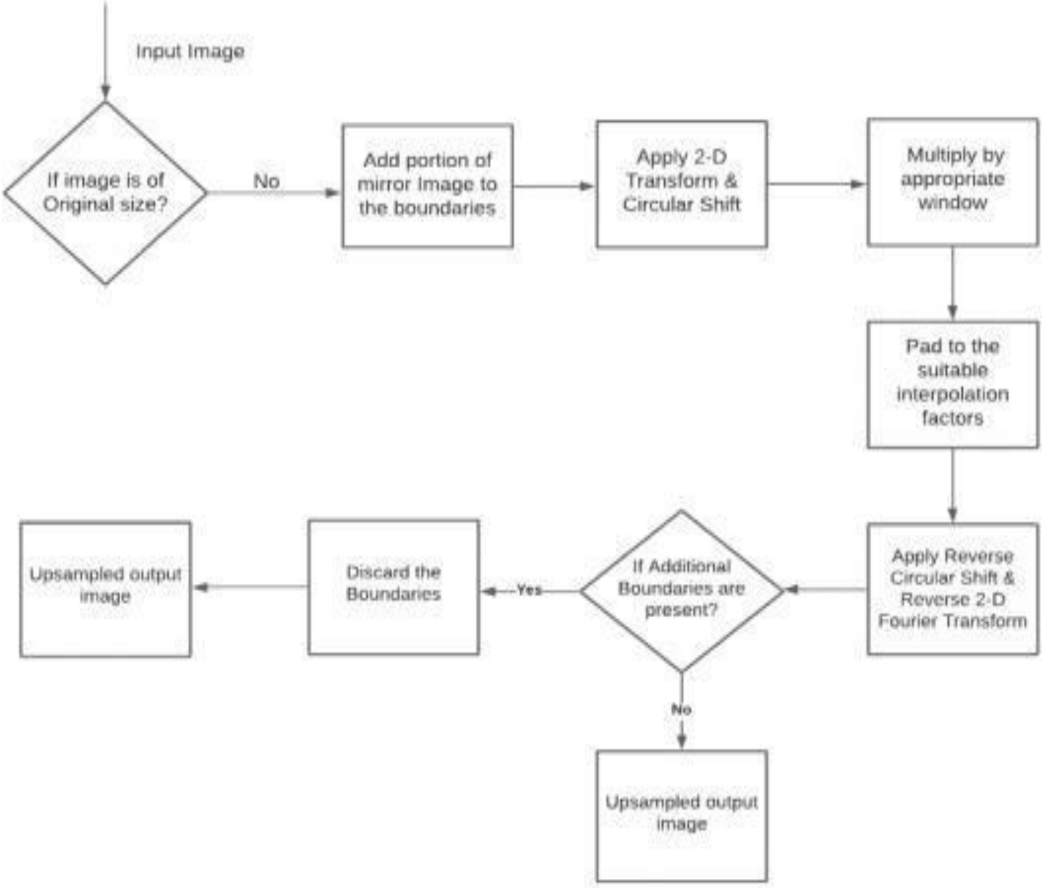
Input Image
If image is of Original size?
No
Add portion of mirror Image to the boundaries
Apply 2-D Transform & Circular Shift
Multiply by appropriate window
Pad to the suitable interpolation factors
Apply Reverse Circular Shift & Reverse 2-D Fourier Transform
If Additional Boundaries are present?
Yes
Discard the Boundaries
Upsampled output image
No
Upsampled output image

Fig.7. Flowchart for Up-sampling Module

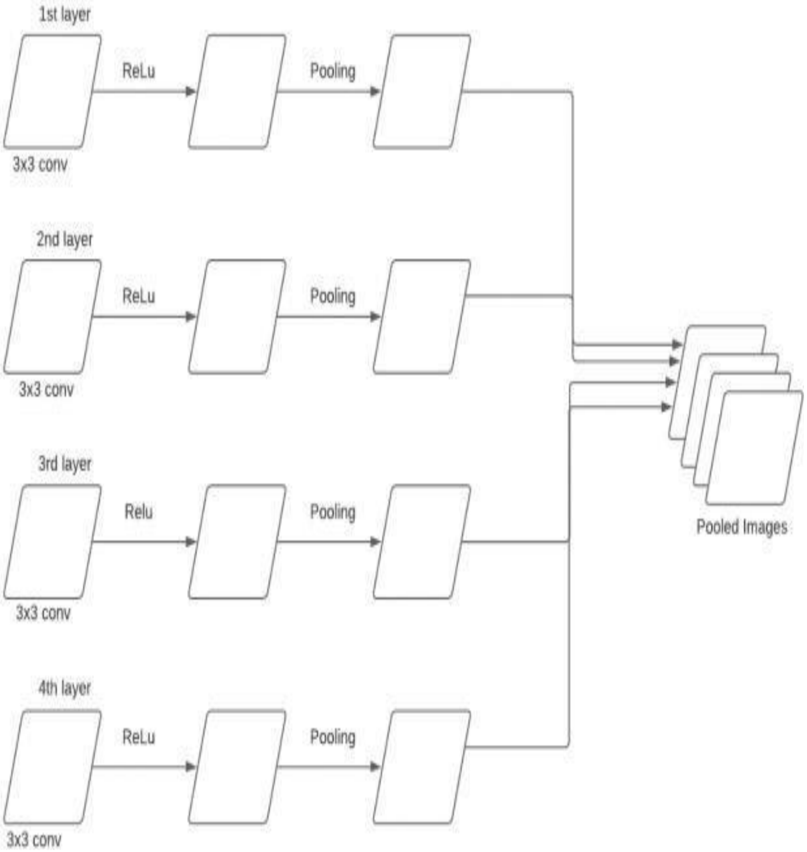

Fig.8. Flowchart for Pooling Module

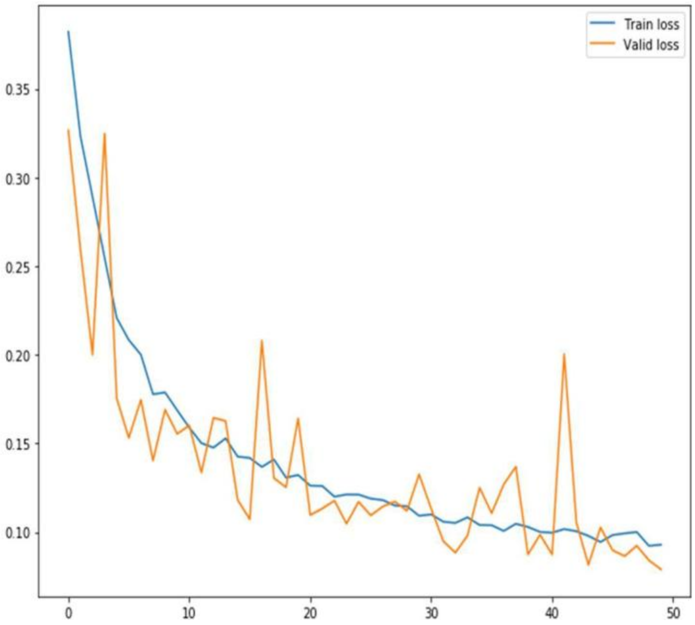

Fig.9 Train loss vs valid loss over epoch

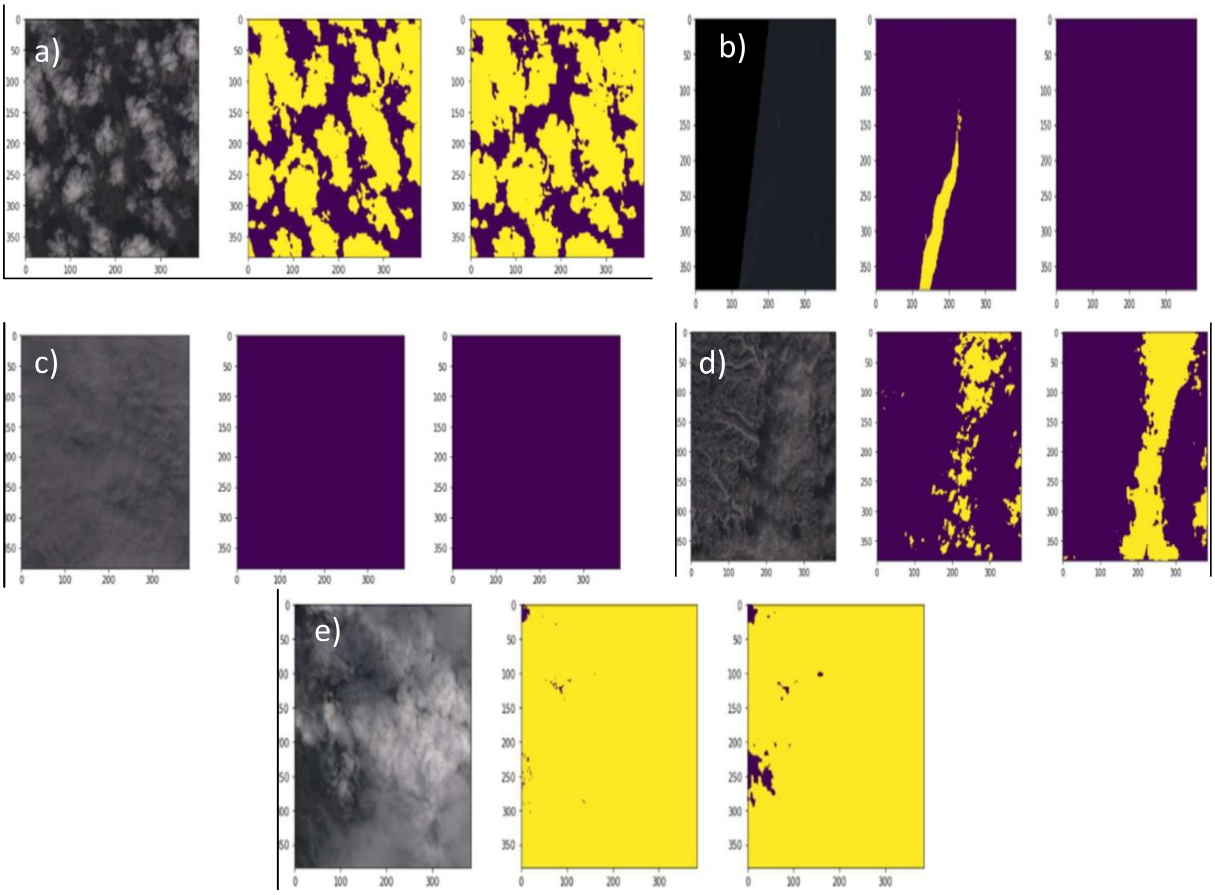

Figure 10: a) gives prediction scenario 1, b) is prediction scenario 2, c) is prediction scenario 3, d) and e) are prediction scenario 4